공학박사 학위논문

Robust Feature Learning with Deep Neural Networks

깊은 신경망을 이용한 강인한 특징 학습

2016년 8월

서울대학교 대학원

전기·컴퓨터공학부

이 태 훈

Robust Feature Learning with Deep Neural Networks

지도교수 윤 성 로

이 논문을 공학박사 학위논문으로 제출함

2016년 5월

서울대학교 대학원

전기·컴퓨터공학부

이 태 훈

이태훈의 박사 학위논문을 인준함

2016년 5월

위 원 장	성 원 용	(인)
부위원장	윤 성 로	(인)
위 원	임 요 한	(인)
위 원	우 경 구	(인)
위 원	원 중 호	(인)

Abstract

Recent advances in machine learning continue to bring us closer to artificial intelligence. In particular, deep learning plays a key role in cutting-edge frameworks such as autonomous driving and game playing. Deep learning refers to a class of multi-layered neural networks, which is rapidly evolving as the amount of data increases, prior knowledge builds up, efficient training schemes are being developed, and high-end hardwares are being build. Currently, deep learning is a state-of-the-art technique for most recognition tasks.

As deep neural networks learn many parameters, there has been a variety of attempts to obtain reasonable solutions over a wide search space. In this dissertation, three issues in deep learning are discussed and approaches to solve them with regularization techniques are suggested. First, deep neural networks expose the problem of intrinsic blind spots called adversarial perturbations. Thus, we must construct neural networks that resist the directions of adversarial perturbations by introducing an explicit loss term to minimize the differences between the original and adversarial samples. Second, training restricted Boltzmann machines show limited performance when handling minority samples in class-imbalanced datasets. Our approach addresses this limitation and is combined with a new regularization concept for datasets that have categorical features. Lastly, insufficient data handling is required to be more sophisticated when deep networks learn numerous parameters. Given high-dimensional samples, we must augment datasets with adequate prior knowledge to estimate a high-dimensional distribution.

Furthermore, this dissertation shows the first application of deep belief networks to identifying junction splicing signals. Junction prediction is one of the major problems in the field of bioinformatics, and is a starting point to understanding the entire gene expression process. In summary, this dissertation proposes a set of deep learning regularization schemes that can learn the meaningful representation underlying large-scale genomic datasets and image datasets. The effectiveness of these methods was confirmed with a number of experimental studies.

Keywords: machine learning, deep learning, manifold learning, deep neural networks, convolutional neural networks, restricted Boltzmann machines, regularization, bioinformatics, splice junction prediction, boosting, class imbalance, biomedical imaging, data augmentation

Student Number: 2012-30945

초 록

최근 기계 학습의 발전으로 인공 지능은 우리에게 한 걸음 더 가까이 다가오게 되었다. 특히 자율 주행이나 게임 플레이 등 최신 인공 지능 프레임워크들에 있어서, 딥 러닝이 중요한 역할을 하고 있는 상황이다. 딥 러닝이란 multi-layered neural networks 과 관련된 기술들을 총칭하는 용어로서, 데이터의 양이 급속하게 증가하며, 사전 지식들이 축적되고, 효율적인 학습 알고리즘들이 개발되며, 고급 하드웨어들이 만들어짐에 따라 빠르게 변화하고 있다. 현재 딥 러닝은 대부분의 인식 문제에서 최첨단 기술로 활용되고 있다.

여러 레이어로 구성된 깊은 신경망은 많은 양의 파라미터를 학습하기 때문에, 방대한 파라미터 집합 속에서 좋은 해를 효율적으로 찾아내는 것이 중요하다. 본 논문에서는 깊은 신경망의 세 가지 이슈에 대해 접근하며, 그것들을 해결하기 위한 regularization 기법들을 제안한다. 첫째로, 신경망 구조는 adversarial perturbations 이라는 내재적인 blind spots 들에 많이 노출되어 있다. 이러한 adversarial perturbations 에 강인한 신경망을 만들기 위하여, 학습 샘플과 그것의 adversarial perturbations 와의 차이를 최소화하는 manifold loss term을 목적 함수에 추가하였다. 둘째로, restricted Boltzmann machines 의 학습에 있어서, 상대적으로 작은 크기를 가지는 클래스를 학습하는 데에 기존의 contrastive divergence 알고리즘은 한계점을 가지고 있었다. 본 논문에서는 작은 클래스에 더 높은 학습 가중치를 부여하는 boosting 개념과 categorical features를 가진 데이터에 적합한 새로운 regularization 기법을 조합하여 기존의 한계점에 접근하였다. 마지막으로, 신경망의 파라미터를 학습하기에 충분하지 않은 데이터가 주어진 경우, 더 정교한 data augmentation

기법을 다룬다. 샘플의 차원이 많을수록, 데이터 생성의 기저에 깔려있는 사전 지식을 활용하여 augmentation을 하는 것이 더욱 더 필요하다.

나아가, 본 논문은 junction splicing signals 학습을 위한 첫 번째 깊은 신경망 모델링 결과를 제시하고 있다. Junction prediction 문제는 positive 샘플 수가 매우 적어 패턴 모델링이 힘들며, 이는 생명정보학 분야에서 가장 중요한 문제 중 하나로서, 전체 gene expression process 를 이해하는 첫 걸음이라고 할 수 있다. 요약하면, 본 논문은 딥 러닝으로 이미지와 대용량 유전체 데이터를 위한 효과적인 표현법을 학습할 수 있는 regularization 기법들을 제안하였으며, 유명한 벤치마크 데이터와 biomedical imaging 데이터를 사용하여 그 실효성을 검증하였다.

주요어: machine learning, deep learning, manifold learning, deep neural networks, convolutional neural networks, restricted Boltzmann machines, regularization, bioinformatics, splice junction prediction, boosting, class imbalance, biomedical imaging, data augmentation

학번: 2012-30945

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Deep neural networks

Deep networks have been adopted successfully for various tasks, such as object recognition [54], face recognition [82], question answering [76], autonomous driving [14], and game playing [71, 92]. Applications in bioinformatics include protein structure prediction [61], drug-target interaction prediction [111], and tissue-regulated splicing prediction at the RNA level using RNA-seq data [62].

Deep learning refers to a large family of machine learning methods that learn hierarchical representations of data. Traditional learning frameworks usually consist of two steps: the extraction of handcrafted features and the construction of learning models. For example, in image recognition, object classification has been performed with well-known feature descriptors such as the scale-invariant feature transform (SIFT) and the histogram of oriented gradients (HOG) etc. However, use of these features usually relies on expert knowledge and may not generalize well. One of the purposes of deep learning is to replace handcrafted features with data-driven hierarchical features using

rich data and efficient algorithms.

By adding more hidden units or layers, we can learn more complicated relationships between sensory data and desired outputs. However, increasing model complexity incurs an enormous solution space. Thus, regularization techniques have been combined with deep neural networks to obtain acceptable solutions.

Deep networks typically cannot be trained well using the conventional back-propagation algorithm [88]; thus, alternative approaches have been proposed [94, 58]. The most popular technique is layer-wise pre-training followed by fine-tuning [39] (considered the first breakthrough in deep learning). In this approach, a multi-layer neural network can be constructed by stacking well-trained two-layer models and feeding the outputs of the previous layer into the inputs of the next layer. To train the two-layer sub-block, we can utilize an auto-encoder [8] or a restricted Boltzmann machine (RBM) [37]. A multi-layer network based on the former is called a *stacked auto-encoder* [108]. A network composed of the latter is referred to as a *deep belief network* (DBN) [39].

There are a huge number of model variants and different training algorithms in the deep learning field. However, most of them have been developed by extending base architectures, such as convolutional or recurrent neural networks. Because the field of deep learning is fast-evolving, there are no standard network architectures for any specific task (*e.g.*, a large number of model variants appear for the ImageNet competition every year). Thus, we must conduct studies that consider general-purpose prior knowledge. This dissertation can be expected to contribute to large portion of deep learning algorithms by proposing new regularization techniques.

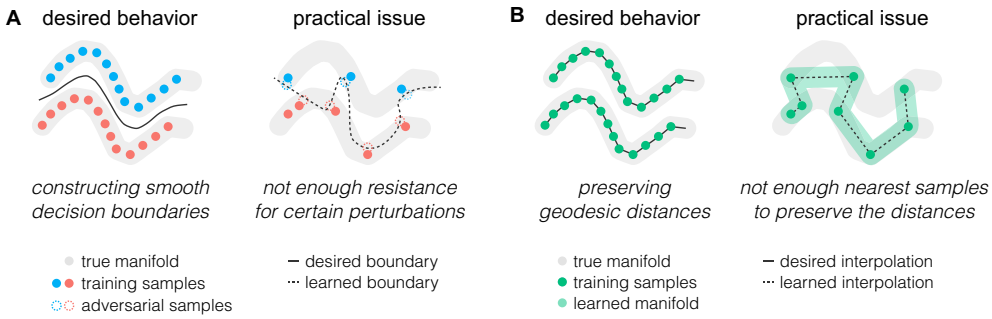

Figure 1.1: Desired behaviors and practical issues of deep learning and manifold learning. (a) Deep learning discriminates different classes; however, it may result in wiggly boundaries vulnerable to adversarial perturbations. (b) Manifold learning preserves geodesic distances; however, it may result in poor embedding.

1.2 Issue 1: adversarial examples handling

Previous regularization techniques involved designing efficient training schemes (*i.e.*, dropout [97], DropConnect [109], and Batch Normalization [44]) and well-structured networks [*i.e.*, Network In Network [63], and Inception [99]]. Even with these cutting-edge techniques, deep neural networks are still prone to performance degradation when certain small perturbations are injected into samples. The perturbations, which are barely perceptible to humans but make neural networks less confident, are called *adversarial examples* [100, 30, 74]. This phenomenon occurs when there are fewer training examples than parameters and the inner products of high-dimensional vectors [30]. For fully connected layers, activation grows by ϵn in the worst-case, when n components of an inner product are changed by ϵ. As shown in Fig. 1.1(a), decision boundaries constructed by deep networks are wiggly and sensitive to adversarial perturbations.

In [30] in particular, adversarial training was proposed to minimize classification loss both on given samples and on adversarial examples. This type of

training helps neural networks to increase their confidence scores even when faced with corrupted samples and generalize across different clean examples. However, the gradients of a discriminative objective function may vanish when the gradients of both the original and adversarial examples are aggregated.

To address this issue, we present a new network called the *manifold regularized network (MRnet)*, which regularizes deep neural networks based on the concept of manifold learning. Manifold learning refers to methodologies based on the manifold hypothesis, in which the nearest samples in a high-dimensional input space are also the nearest pairs on a manifold of much lower dimensionality. Traditional manifold learning [*i.e.*, ISOMAP [101], Locally Linear Embedding [87], and t-SNE [106]] techniques are well-formulated to find a transformation preserving geodesic distances in high-dimensional space. However, they demonstrate limited performance in practice because there are insufficient nearest training samples (Fig. 1.1(b)). There have been many attempts [82, 103, 115] to unify deep learning and manifold learning. These studies have common limitations inherited from those of manifold learning.

In this paper, we generate adversarial examples and make neural networks insensitive to the direction of adversarial perturbations by adding a manifold penalty term. Similar to popular deep learning techniques, the manifold penalty can be easily applied to gradient-based optimization. We then demonstrate that our method can find appropriate manifold representations for our three tested benchmark datasets.

1.3 Issue 2: class-imbalance handling

To handle class-imbalance, we propose a new machine learning method. We develop a new training algorithm for restricted Boltzmann machines (RBMs) and demonstrate its efficiency for predicting splicing patterns. A deep belief network (DBN, [39]) learns high-level features from unlabeled data [38] and then fine-tunes the weights to improve discriminative performance. Leveraged by this two-step learning, a DBN has strong generalization abilities, which has resulted in breakthroughs for various applications [23].

Our proposal includes a novel procedure for training restricted Boltzmann machines (RBMs) comprising a DBN. This procedure improves contrastive divergence (CD, [38]) when training an RBM for DNA sequences, and more generally, for binary representations of categorical information. The basic idea of our approach is similar to that of boosting in ensemble learning. We thus name the proposed training scheme *boosted CD with categorical gradient*.

In our experiments, the approach achieved F1-scores nearly 20% higher than the alternatives and was particularly effective for training in class imbalanced problems. We present our work in the context of genomics, but it is applicable to learning for other types of data that contain categorical features (*i.e.*, text mining).

1.4 Issue 3: insufficient data handling

While there is plenty of data for natural image processing (*i.e.*, 1.2M of ImageNet [89]), there is a relatively small amount of data available for biomedical image processing and CNN training. To aid in the recognition of comets with CNN, we augmented individual comet images due to the insufficient amount

of data, which is one of the major issues in biomedical image processing.

Here, we propose a deep learning based automated tool for high-throughput comet-assay analysis called *DeepComet*. Damage to DNA can result in cancer, aging, and other serious diseases. The comet assay is sensitive to multiple types of DNA lesions and has been used in various applications. Although recent comet assay platforms have improved the limited throughput and reproducibility of traditional assays, analyzing comet data in large quantities often demands tremendous human effort. To alleviate this, we propose DeepComet, a computational tool that can rapidly characterize a large number of comets and classify comet images based on convolutional neural networks.

We test DeepComet using real-world data from 35 high-throughput comet assay experiments, giving over 700 comets in total. According to our results, the proposed method provides unprecedented levels of performance as an automated comet recognition tool in terms of robustness (measured by precision and recall) and throughput. It is our hope that DeepComet can significantly facilitate the entire high-throughput comet-assay analysis process by accelerating the most rate-limiting step. An online implementation of DeepComet is freely available at http://147.46.126.127:11080/comet/.

1.5 Organization

This dissertation is organized as follows: In chapter 2, we review the background necessary to develop the proposed method. In chapter 3, we discuss the unification and similarities between deep and manifold learning. To overcome the problem presented by adversarial perturbations on deep learning, we incorporate an explicit loss term to preserve neighborhood relationships

in deep neural networks, also called manifold regularized networks (*MRnet*). For a smart interpolation in manifold space, we generate adversarial examples that are sufficiently close to the original samples on the manifolds. Results from experiments on the proposed approach and other previous methods are compared in order to validate the efficiency of MRnet. The contents of this chapter are based on the research of

- Taehoon Lee, Minsuk Choi, and Sungroh Yoon, "Manifold Regularized Deep Neural Networks using Adversarial Examples," *arXiv preprint*, arXiv:1511.06381, 2015.

In chapter 4, we propose a new machine learning method for predicting splicing patterns using a deep belief network (DBN, [39]). This procedure improves contrastive divergence (CD, [38]) when training an RBM for DNA sequences. More generally, it also improves binary representations of categorical information. The general concept of our approach is similar to that of boosting in ensemble learning. We thus name the proposed training scheme *boosted CD with categorical gradient*. We present our work in the context of genomics, but it is applicable to learning from other types of data that contain categorical features. The contents of this chapter are based on the following research:

- Taehoon Lee and Sungroh Yoon, "Boosted Categorical Restricted Boltzmann Machine for Computational Prediction of Splice Junctions," in *Proceedings of International Conference on Machine Learning (ICML)*, Lille, France, July 2015.

- Byunghan Lee, Taehoon Lee, and Sungroh Yoon, "DNA-Level Splice Junction Prediction using Deep Recurrent Neural Networks," in *Proceed-*

ings of NIPS Workshop on Machine Learning in Computational Biology, Montreal, Canada, December 2015.

In chapter 5, we propose an automated tool, based on deep learning, for high-throughput comet-assay analysis called *DeepComet*. The key features of DeepComet include the following. First, DeepComet automatically recognizes individual comets from the input image without making any assumptions about the number of comets or their locations. This is critical in order to reduce the time demands of analyzing high-throughput assays containing many comets. Second, DeepComet is able to detect overlapping comets and isolate them. Without this feature, researchers have no other choice but to discard overlapping comets, despite that the front comets are still eligible for analysis. Because overlaps occur frequently in typical high-throughput comet assays, this functionality is useful in maintaining sufficient comet counts for analysis through salvaging parts of overlapping comets. Third, DeepComet can characterize each recognized comet using convolutional neural networks (CNN) and then report key parameters, such as tail moments, without making overly simplified assumptions on comet shapes, as certain existing tools do. To recognize comets with CNN, we augmented individual comet images because of a lack of sufficient data, one of the major issues in biomedical image processing. Given the effectiveness of DeepComet, it is our hope that DeepComet can facilitate high-throughput comet-assay analysis by accelerating the most rate-limiting steps. The contents of this chapter are based on the following research:

- Taehoon Lee, Sungmin Lee, Woo Young Sim, Yu Mi Jung, Sunmi Han, Joong-Ho Won, Hyeyoung Min, and Sungroh Yoon, "DeepComet: Deep Learning-Based Comet Analysis for DNA Damage Assessment Studies" *(under review)*

In chapter 6, we conclude the dissertation and discusses its contributions towards resolving research issues in robust feature learning with deep neural networks.

Chapter 2

Background

2.1 Basic operations for deep networks

Deep networks are multi-layer neural networks. In this section, the basic operations and notations for deep networks are described. The starting point of a deep neural network is single output node as shown in Fig. 2.1(a). The output node y_1 is computed as a linear combination of multiple input nodes $x_1, \ldots, x_P$ and their weights $w_{11}, \ldots, w_{1P}$. Individual input nodes correspond to variables. For example, in image recognition, each image is single sample

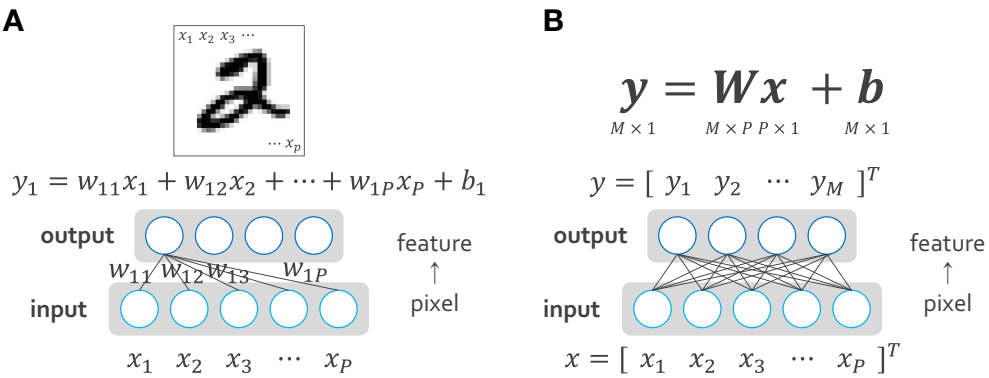

Figure 2.1: Basic operations for single-layer neural networks.

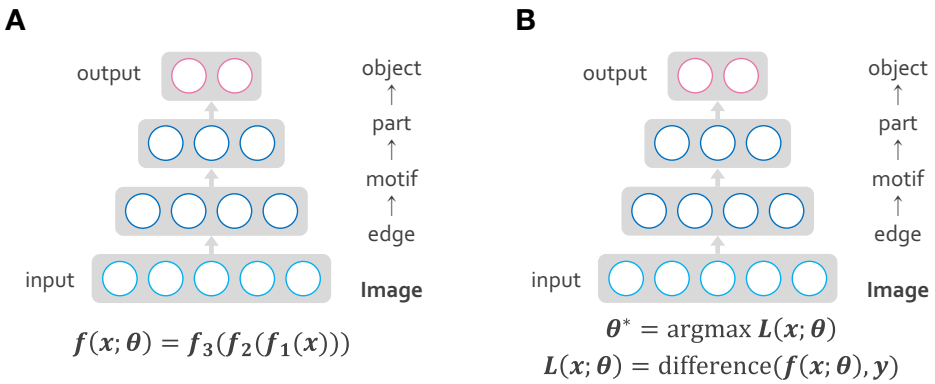

Figure 2.2: Basic operations for multi-layer neural networks.

consisting of P pixels. The intensity of each pixel is mapped to a single input node x_i. Thus, the neuron will combine weighted inputs $\sum_i x_i w_{1i}$ and use them to determine the output. This behavior closely emulates closely our understanding of how real neurons work.

In most literature, vector forms (Fig. 2.1(b)) are preferred over scalar forms (Fig. 2.1(a)) for concise descriptions. We present all variables (pixels) of a sample (image) as a standard column vector $x \in \mathbf{R}^P$. Then, y_1 is defined as $w_1^T x$ where $w_1 \in \mathbf{R}^P$. With this notation, we can compute $y_i = w_i^T x$ for all i simultaneous as $y = Wx$ where $W \in \mathbf{R}^{M \times P}$.

A multi-layer neural network is formed by hooking together many simple output nodes as illustrated in Fig. 2.1, so that the output of one neuron is the input of another. For example, Fig. 2.2 shows a three-layer neural network. Our neural network has parameters $\theta = (W^{(1)}, b^{(1)}, W^{(2)}, b^{(2)}, W^{(3)}, b^{(3)})$ where $W^{(l)}$ denotes the parameter (or weight) associated with the connection between layer $l-1$ and l. In addition, $b^{(l)}$ is the bias associated with layer $l-1$ and l; f_l denotes the activation function in layer l.

Thus, in our example, the final outputs are $f(x;\theta) = f_3(f_2(f_1(x)))$. Note that bias units do not have inputs or connections going into them, since they

always output the value +1. We call this operation forward propagation. In the example, we define the cost function using the forward function with respect to single sample:

$$L(x;\theta) = \text{difference}(f(x;\theta), y).$$

In a supervised setting, the cross entropy is one of the most popular difference measurements between inferred outputs $f(x;\theta)$ and desired outputs y. In an unsupervised setting, the desired output y is usually defined as the original input x and the difference is measured using the 2-norm difference.

2.2 History of deep networks

Deep neural networks originated from traditional artificial neural networks. Fig. 2.3 shows a brief history of artificial neural networks. Beginning with Perceptron [84], early models such as Cognitron [27] and NeoCognitron [26] were proposed in the 1960-70's. In 1971, a deep network consisting of 8 layers had been already described [45]. However, their performance was limited in practice due to the small amount of available data and simple training algorithms.

After the advent of the back-propagation algorithm [37] in the mid-80s, artificial neural networks with multiple adaptive hidden layers were briefly revisited. However, many researchers again abandoned neural networks in the 1990's because support vector machines (SVM) [19] had better performance. A SVM is a clever type of perceptron developed by Vapnik and his co-workers in 1995. A decade later, in 2006, a publication [39] by Geoffrey Hinton and Ruslan Salakhutdinov drew additional attention to neural networks by showing that many-layered feedforward neural networks could be effectively pre-trained one

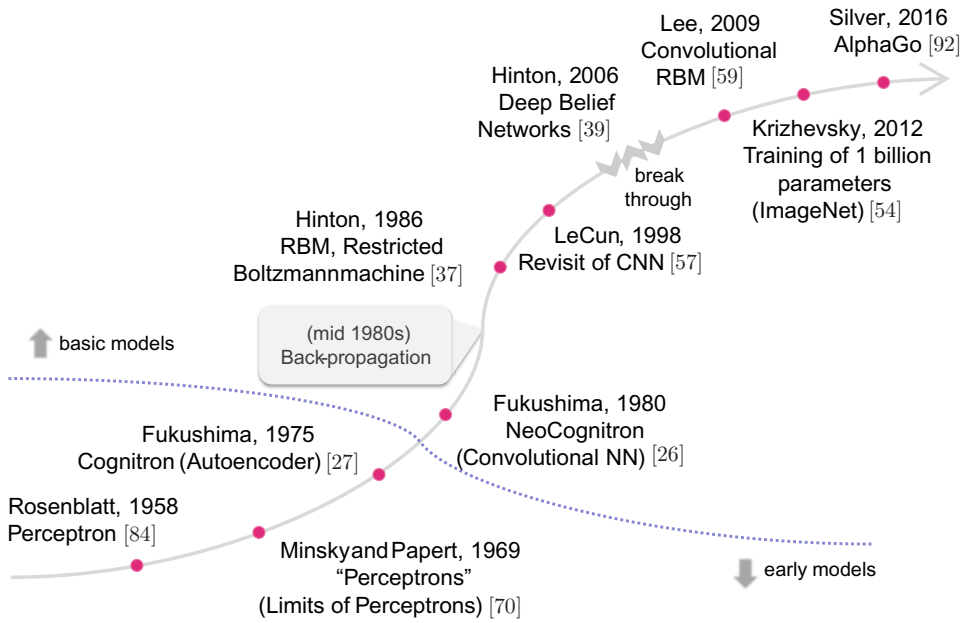

Figure 2.3: History of artificial neural networks.

layer at a time. This was done by treating each layer in turn as an unsupervised restricted Boltzmann machine, then fine-tuning the network using standard backpropagation [37]. This was hailed as breakthrough for artificial neural networks by many researchers.

Since the breakthrough, deep learning has become part of many state-of-the-art systems in various disciplines, particularly computer vision and speech recognition. Results on commonly used evaluation sets such as MNIST [57] (hand-written digits image classification), ILSVRC [89] (natural image classification) and TIMIT [28] (speech recognition), as well as a range of large-vocabulary speech recognition tasks are constantly being improved on with new deep learning applications. Advances in hardware have also been instrumental in the renewed interest in deep learning. Specifically, powerful graphics processing units (GPUs) are well-suited for the matrix-vector operations involved in machine learning. GPUs have been shown to speed up training

algorithms by several orders of magnitude, bringing running times from weeks back to days.

2.3 Modern deep networks

Modern deep networks have three major connectivities: fully-connected, convolutional, and recurrent connections. The connectivity of neural networks refers to the connection type between two layers and involves prior knowledge of each field as shown in Fig. 2.4. Convolutional layers are appropriate for data that have 2D structures (*spatial dependency*) like images. Convolutional neural networks have become the method of choice for processing visual and other two-dimensional data. Convolutional networks may include local or global pooling layers, which combine the outputs of neuron clusters. They also consist of various combinations of convolutional and fully connected layers, with point-wise nonlinearity applied at the end of each layer. To reduce the number of free parameters and improve generalization, a convolution operation on small regions of input is introduced. One major advantage of convolutional networks is the use of shared weights in convolutional layers, which means that the same filter (weight bank) is used for each pixel in the layer. This both reduces memory footprint and improves performance.

Recurrent layers are suitable for sequential data where variables have a temporal order (*temporal dependency*), such as audio and natural languages. Numerous researchers now use variants of recurrent neural networks called the Long Short-Term Memory (LSTM) network [41] and the gated recurrent unit (GRU) [16]. LSTM and GRU are augmented by additional gates units such as forget gates and recurrent gates. Compared to a basic form of recurrent neural

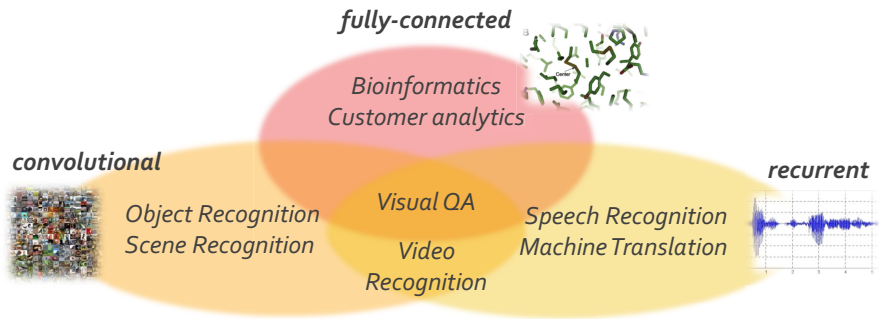

Figure 2.4: Three major connectivities of artificial neural networks. The connectivity of neural networks involves prior knowledge of each field.

networks, LSTM and GRU are able to solve complex learning tasks requiring memorization of events. For other applications, such as gene expression and customer analytics, fully connected layers are popular due to weak domain knowledge.

In addition to the three major types of layers, new structures are being continuously proposed (*e.g.*, external long-term memory [113] and external random-access memory [55]). The latest applications (see Fig. 2.5) incorporate many of these concepts. For example, memory networks [113] have been successfully applied in the context of question answering (QA) where long-term memory effectively acts as a dynamic knowledge, and the output is a text response. Visual QA is derived from both scene understanding and natural language understanding. In [76], CNNs and RNNs were utilized together to recognize image and text. DeepMind shows playing Atari video games using deep reinforcement learning [71]; AlphaGo [92] achieved one of the long-standing grand challenges of AI by learning the game of Go well enough to beat a professional human Go player.

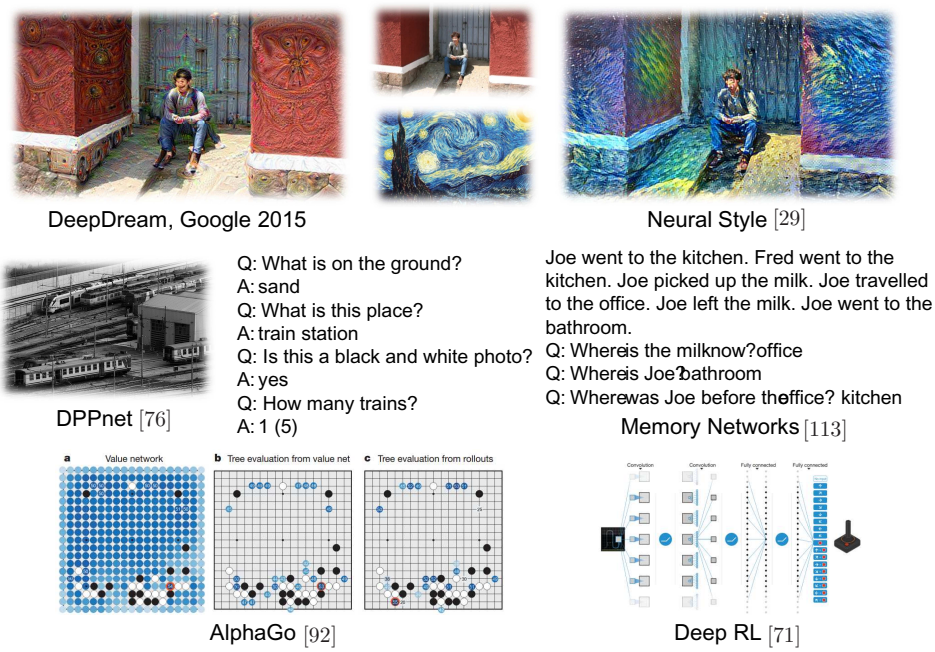

Figure 2.5: Latest applications of artificial neural networks.

2.3.1 Contrastive divergence

A restricted Boltzmann machine contains stochastic binary-valued hidden units $\mathbf{h} = \{h_1, ..., h_{n_h}\}$ and visible units $\mathbf{v} = \{v_1, ..., v_{n_v}\}$, where n_h and n_v are the numbers of hidden and visible units, respectively. As shown in Fig. 2.6(a), the two layers $\mathbf{v}$ and $\mathbf{h}$ are tied with symmetrically weighted connections denoted by W, forming a bipartite graph. W is represented by an $n_v \times n_h$ matrix, where w_{ij} is the weight for the connection between v_i and h_j.

Each hidden node has an activation probability given by $P(h_j = 1|\mathbf{v}) = \text{sigm}(c_j + \sum_{i=1}^{n_v} v_i w_{ij})$, where the activation function $\text{sigm}(x) = 1/(1+\exp(-x))$ is shown in Fig. 2.6(b) and c_j is the bias weight for the hidden unit h_j. Similarly, the activation probability of visible unit v_i is given by $P(v_i = 1|\mathbf{h}) = \text{sigm}(b_i + \sum_{j=1}^{n_h} w_{ij} h_j)$, where b_i is the bias unit for the visible unit v_i. The

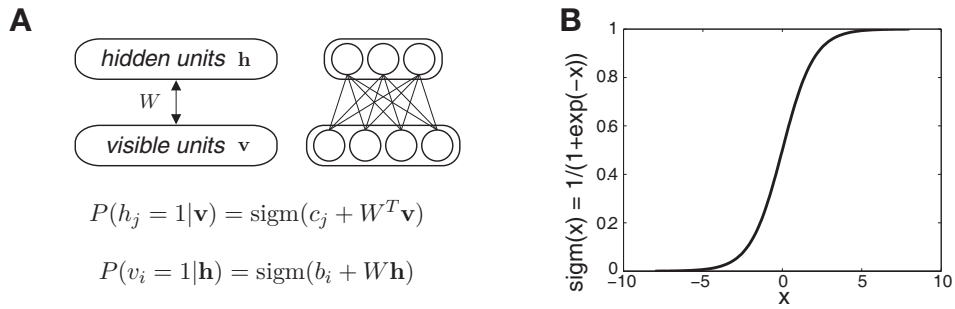

Figure 2.6: RBM representation and sigmoid activation.

energy of the network can be defined as

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{n_v} b_i v_i - \sum_{j=1}^{n_h} c_j h_j - \sum_{i=1}^{n_v}\sum_{j=1}^{n_h} v_i w_{ij} h_j. \tag{2.1}$$

The joint probability for two vectors $\mathbf{v}$ and $\mathbf{h}$ is given by $P(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})}$ where the normalization constant $Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$. The probability of observing a set of visible units $\mathbf{v}$ is given by $P(\mathbf{v}) = \frac{1}{Z}\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}$. We can formulate an optimization problem for training RBM to minimize the negative log-likelihood (NLL) of data.

When the gradient descent is utilized to minimize the negative log-likelihood, the CD procedure is normally used to approximate the gradient due to the intractable Z inside $P(\mathbf{v})$. The derivative of the NLL with respect to w_{ij} is

$$\begin{aligned}\frac{\partial L}{\partial w_{ij}} &= \frac{\partial}{\partial w_{ij}}\mathbf{E}\left[-\log\left(\frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v}_n,\mathbf{h})}}{\sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}\right)\right] \\ &= \mathbf{E}_{data}[v_i h_j] - \mathbf{E}_{model}[v_i h_j].\end{aligned}$$

While $\mathbf{E}_{data}[v_i h_j]$ is a simple expectation over observed vectors, $\mathbf{E}_{model}[v_i h_j]$ requires the probabilities of all possible visible states.

To approximate the expectation, CD exploits Gibbs sampling [56], which

enables the estimation of the distribution using samples close to observations. The derivative from k-step Gibbs sampling (see Fig. 4.1) is

$$\frac{\partial L}{\partial W} \approx \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{v}_n^{(0)} \mathbf{h}_n^{(0)^T} - \mathbf{v}_n^{(k)} \mathbf{h}_n^{(k)^T} \right). \tag{2.2}$$

In the same manner, the derivatives of $\mathbf{b}$ and $\mathbf{c}$ can be approximated as follows [5]:

$$\frac{\partial L}{\partial \mathbf{b}} \approx \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{v}_n^{(0)} - \mathbf{v}_n^{(k)} \right), \tag{2.3}$$

$$\frac{\partial L}{\partial \mathbf{c}} \approx \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{h}_n^{(0)} - \mathbf{h}_n^{(k)} \right). \tag{2.4}$$

2.3.2 Deep manifold learning

Despite the recent success of deep networks, their ultimate goal has not yet been reached, which is generalizing lower-dimensional manifolds. Researchers anticipate that neural networks with multiple layers could learn a manifold embedding and classifier simultaneously [6]. However, traditional loss terms, such as a reconstruction or classification error, are not sufficient to capture local variations on manifolds. To support the manifold hypothesis, we need to employ another type of cost function that makes neighborhoods have a similar representation. Hence, there have been many attempts to combine both functional concepts of deep and manifold learning.

Attempts to unify deep and manifold learning can be divided into two categories: manifold learning-based unifying and deep learning-based unifying. The former finds a hierarchical manifold embedding with layer-wise manifold learning in the same way as pre-training in deep belief networks. For example, Locally Linear Embedding [87] was used as a base unit for a deep architec-

ture [115]. The latter incorporates an additional objective from the manifold learning perspective into a framework of deep learning. For instance, the following objective was proposed in [82]:

$$\sum_{(h_i,h_j)\in\mathcal{D}_{sim}} \|h_i - h_j\|_2^2 + \sum_{(h_i,h_k)\in\mathcal{D}_{dis}} \max(0, \beta - \|h_i - h_k\|_2)^2, \tag{2.5}$$

where $\{h_1, \ldots, h_n\}$ is a set of hidden representations obtained from forward operations of deep neural networks. Two sets $\mathcal{D}_{sim}$ and $\mathcal{D}_{dis}$ denote the sets of data pairs with the same labels and different labels, respectively. Another form is defined in [103] as follows: $\sum_{\mathcal{D}_{sim}} w_{ij}\|h_i - h_j\|^2 - \sum_{\mathcal{D}_{dis}} w_{ik}\|h_i - h_k\|^2$ where $w_{ij} = \exp(-\|x_i - x_j\|^2/\rho)$. Other variations [*i.e.*, [65]] are also possible; however, variations still have limitations inherited from the original manifold learning. Training set neighborhood information may be problematic because most nearest samples have too little in common in high-dimensional Euclidean spaces [6].

Chapter 3

Adversarial examples handling

3.1 Introduction

In our study, we incorporate an explicit loss term to preserve neighborhood relationships into deep neural networks. As mentioned earlier, neighborhood information based on only training samples may cause inappropriate embedding. Because the samples are sufficiently densely populated in most cases, we generate adversarial examples that are sufficiently closed to original samples on manifolds. An example of an adversarial perturbation is shown in Fig. 3.1 [retrieved from [30]].

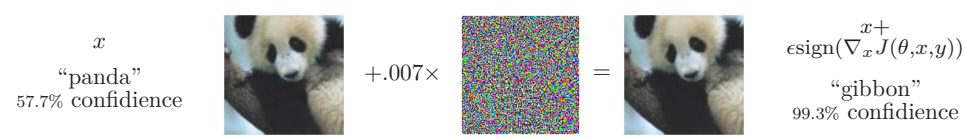

Figure 3.1: An example of adversarial perturbation applied to GoogLeNet [99] [retrieved from [30]].

We can obtain these adversarial perturbations easily by updating a sample instead of parameters. While the parameter updating performs $\theta := \theta - \eta \nabla_\theta J(\theta; x, y)$, the sample updating calculates $x := x + \eta \nabla_x J(\theta; x, y)$, where η is a learning rate. The opposite signs in the two update equations mean that the former minimizes a classification loss but the latter maximizes a classification loss. Given a panda image with a high confidence score (see Fig. 3.1), the direction can be calculated by making the neural network less confident. We can produce a new image that a human cannot distinguish from the original, but the neural network believes the image is a gibbon with 99.3% confidence.

The problem of adversarial perturbations arises in several learning models as well as state-of-the-art deep networks, such as AlexNet [54] or GoogLeNet [99]. An ensemble of different deep architectures trained on different subsets of the training data also misclassify the same adversarial example. This suggests that adversarial examples expose fundamental blind spots in our objective functions [100]. Thus, we need to make neural networks resist directions of adversarial perturbations by introducing an explicit loss term to minimize differences between original and adversarial samples.

3.2 Methods

3.2.1 Manifold regularized networks

Given a data set $\mathcal{X} = \{x_1, \ldots, x_n\}$ with corresponding labels $\mathbf{y} = \{y_1, \ldots, y_n\}$, we consider the following objective loss function:

$$J(\theta; \mathcal{X}, \mathbf{y}) = L(\theta; \mathcal{X}, \mathbf{y}) + \lambda \Omega(\theta) \tag{3.1}$$

where $L(\cdot)$ is a classification loss and $\Omega(\cdot)$ is defined to maximize a prior $p(\theta)$. For the two terms $L(\cdot)$ and $\Omega(\cdot)$, the cross entropy and L2 weight decay are most popular choices in a supervised setting. L2 decay works well in practice; however, these forms of neural networks have intrinsic blind spots due to the huge number of parameters and linear functions using them. Hence, we propose an additional manifold loss term that exploits the characteristics of blind spots.

Suppose we have a neural network with $L+1$ layers. The last $(L+1)$-th layer is an softmax layer. Let $l \in \{1, \ldots, L+1\}$ be a layer index and $a^{(l)}$ an activation of the l-th layer ($a^{(1)} = x$). The proposed objective can be defined as:

$$J_m(\theta; \mathcal{X}, \mathbf{y}) = L(\theta; \mathcal{X}, \mathbf{y}) + \lambda\Omega(\theta) + \lambda_m \Phi(\mathcal{X}, \mathcal{X}'), \tag{3.2}$$

$$\Phi(\mathcal{X}, \mathcal{X}') = \frac{1}{n}\sum_n \Phi(x_n, x'_n) = \frac{1}{2n}\sum_n \|a_n^{(L)} - a_n'^{(L)}\|_2^2, \tag{3.3}$$

$$x'_n = x_n + \beta \underbrace{\nabla_{x_n} L(\theta; x_n, y_n) / \|\nabla_{x_n} L(\theta; x_n, y_n)\|}_{\Delta x_n} \tag{3.4}$$

where Φ is a manifold loss and λ_m is a hyper-parameter for the manifold loss. Both $a_n^{(L)}$ and $a_n'^{(L)}$ are the last hidden layer's activations using a standard feed-forward operation. Equation (3.4) denotes the generation of an adversarial example x'_n from x_n.

Fig. 3.2 shows an overview of the proposed methodology. To make a deep neural network robust to adversarial perturbations, we regard the activations of the last hidden layer as a manifold representation and minimize the difference between the two manifold embeddings of x and x' as shown in Fig. 3.2(a). The proposed forward and backward operations to solve (3.2) are presented in

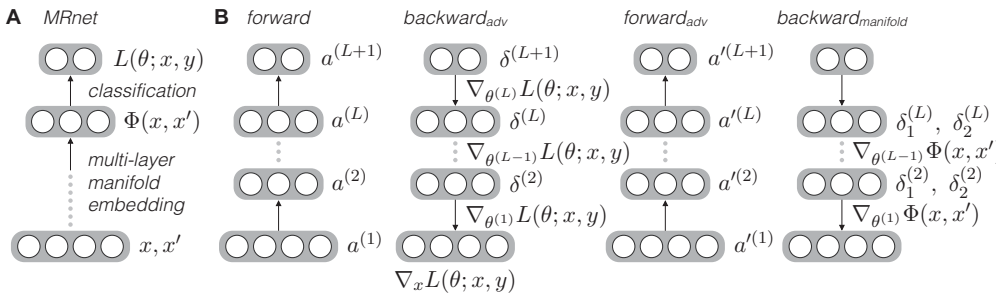

Figure 3.2: (a) The proposed methodology learns both classifier and manifold embedding that is robust for adversarial perturbations. (b) Forward and backward operations of MRnet. The first forward operation is the same as in a standard neural network. The following backward$_{adv}$ is the same as the standard back-propagation except that an adversarial perturbation $\nabla_x L(\theta; x, y)$ is computed in the first layer.

Fig. 3.2(b). The forward operation is the same as in a standard neural network:

$$a^{(l+1)} = f(z^{(l+1)}) = f(W^{(l)} a^{(l)} + b^{(l)}), \qquad \text{if } l\text{-th layer is fully-connected,} \tag{3.5}$$

$$\left[a^{(l+1)}\right]_j = f(\left[z^{(l+1)}\right]_j) = f(\sum_i \left[a^{(l)}\right]_i * W_{ij}^{(l)} + b_j^{(l)}), \qquad \text{if } l\text{-th layer is convolutional} \tag{3.6}$$

where $f(\cdot)$ is an activation function (chosen to be the rectified linear unit in this paper) and $*$ is the convolutional operation of a valid size. For a convolutional layer, $[a^{(l)}]_i$ and $[a^{(l+1)}]_j$ are the i-th and j-th feature map in the l-th layer and $(l+1)$-th layer, respectively. $W_{ij}^{(l)}$ is a convolutional filter between $[a^{(l)}]_i$ and $[a^{(l+1)}]_j$. The following backward$_{adv}$ is also the same as standard back-propagation except that an adversarial perturbation $\nabla_x L(\theta; x, y)$ is computed

in the first layer:

$$\nabla_x L(\theta; x, y) = (\delta^{(2)} \circ g(a^{(2)}))W^{(1)}, \qquad \text{if 1-st layer is fully-connected,} \tag{3.7}$$

$$\Big[\nabla_x L(\theta; x, y)\Big]_i = \sum_j \big([\delta^{(2)}]_j \circ g([a^{(2)}]_j)\big) \star W_{ij}^{(1)}, \qquad \text{if 1-st layer is convolutional} \tag{3.8}$$

where $g(\cdot)$ is the derivative of an activation function, $\star$ is the convolutional operation of the full size, and $\circ$ is the Hadamard product (an element-wise multiplication).

Next, another forward operation is performed to obtain all the activations of an adversarial example $a'^{(1)} = x' = x + \Delta x$ and calculate $\nabla_\theta \Phi(\mathcal{X}, \mathcal{X}')$. Recall that $\Phi(x_n, x'_n)$ is the squared error between the original $a_n^{(L)}$ and its adversarial $a_n'^{(L)}$:

$$\Phi(x_n, x'_n) = (1/2)\|a_n^{(L)} - a_n'^{(L)}\|_2^2 = (1/2)(a_n^{(L)} - a_n'^{(L)})^T(a_n^{(L)} - a_n'^{(L)}). \tag{3.9}$$

Similar to the backpropagation algorithm, we first compute error terms $\delta_1^{(L)}$ and $\delta_2^{(L)}$:

$$\delta_1^{(L)} = -(a'^{(L)} - a^{(L)}), \quad \delta_2^{(L)} = -(a^{(L)} - a'^{(L)}). \tag{3.10}$$

The error $\delta_1^{(L)}$ is the difference of $a'^{(L)}$ with respect to $a^{(L)}$, and the error $\delta_2^{(L)}$ is the difference of $a^{(L)}$ with respect to $a'^{(L)}$. Now, we can present the gradients and back-propagation for the manifold loss term Φ for each layer $l = L, \ldots, 2$.

If the l-th layer is fully-connected, the rules are as follows:

$$\nabla_{W^{(l-1)}}\Phi(\mathcal{X},\mathcal{X}') = \frac{1}{n}\Big((\delta_1^{(l)} \circ g(a^{(l)}))(a^{(l-1)})^T + (\delta_2^{(l)} \circ g(a'^{(l)}))(a'^{(l-1)})^T\Big), \tag{3.11}$$

$$\nabla_{b^{(l-1)}}\Phi(\mathcal{X},\mathcal{X}') = \frac{1}{n}\Big((\delta_1^{(l)} \circ g(a^{(l)}))\mathbf{1} + (\delta_2^{(l)} \circ g(a'^{(l)}))\mathbf{1}\Big), \tag{3.12}$$

$$\delta_1^{(l-1)} = (W^{(l)})^T\delta_1^{(l)}, \quad \delta_2^{(l-1)} = (W^{(l)})^T\delta_2^{(l)}. \tag{3.13}$$

The rules for a convolutional layer are as follows:

$$\Big[\nabla_{W^{(l-1)}}\Phi(\mathcal{X},\mathcal{X}')\Big]_i = \frac{1}{n}\sum_j \Big(a^{(l-1)} * (\delta_1^{(l)} \circ g(a^{(l)})) + a'^{(l-1)} * (\delta_2^{(l)} \circ g(a'^{(l)}))\Big), \tag{3.14}$$

$$\Big[\nabla_{b^{(l-1)}}\Phi(\mathcal{X},\mathcal{X}')\Big]_i = \frac{1}{n}\sum_{row}\sum_{col}\sum_{n}\Big((\delta_1^{(l)} \circ g(a^{(l)})) + (\delta_2^{(l)} \circ g(a'^{(l)}))\Big), \tag{3.15}$$

$$\big[\delta_1^{(l-1)}\big]_i = \sum_j \big[\delta_1^{(l)}\big]_j \star W_{ij}^{(l)}, \tag{3.16}$$

$$\big[\delta_2^{(l-1)}\big]_i = \sum_j \big[\delta_2^{(l)}\big]_j \star W_{ij}^{(l)}. \tag{3.17}$$

3.2.2 Generation of adversarial examples

For the generation of adversarial examples, we used constant-norm perturbation, while the max-norm constrained perturbation was used in the original paper regarding the problems of adversarial examples. In our experiments, the constant-norm and max-norm approaches did not exhibit significant differences. The important aspect was to select a degree of adversarial perturbations β.

Because adversarial examples become actual instances of a different class when β is greater than a certain threshold, we had to select a degree of pertur-

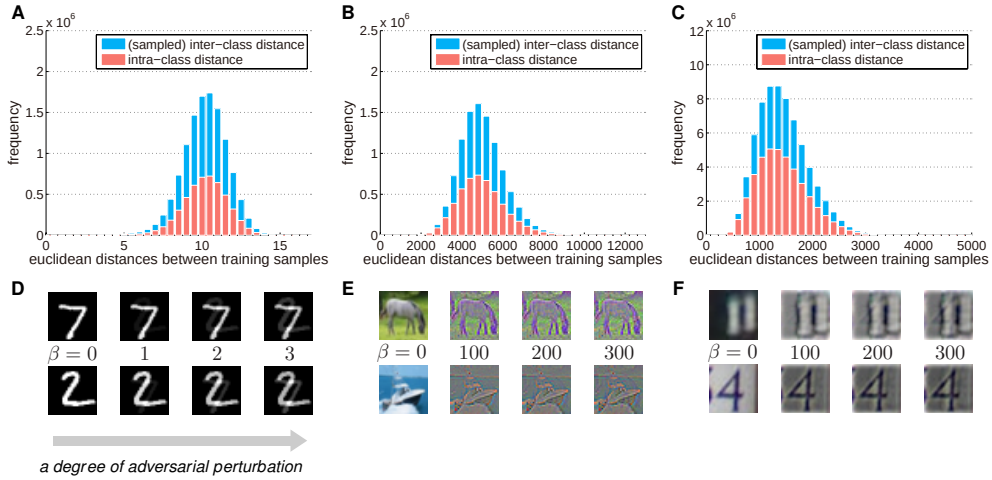

Figure 3.3: (a,b,c) Distributions of Euclidean distances between training samples on individual datasets. (d,e,f) Different perturbation levels on individual datasets. Note that $x' = x + \beta\Delta x$ where Δx is an adversarial perturbation of a unit norm ($\|\Delta x\|_2 = 1$). We chose β in the range that did not violate class information.

bation level β carefully. In Fig. 3.3(a-c), each histogram shows the distribution of pairwise distances. We set β in three datasets to 2, 200, and 200, respectively. These noise levels are small enough to maintain class memberships and the manifold hypothesis. We recommend β in the range of minimum values among intra-class distances.

3.3 Results and discussion

The optimization of the proposed method was conducted using standard stochastic gradient descent. The mini-batch size and the momentum were set to 100 and 0.9, respectively. The learning rate was annealed as described in [109]. For each subsection, we present an initial learning rate and three numbers of epochs, such as 0.001 (100-20-10). We trained models with the initial rate for the first number of epochs. Then, we multiplied by 0.1 for the second epochs

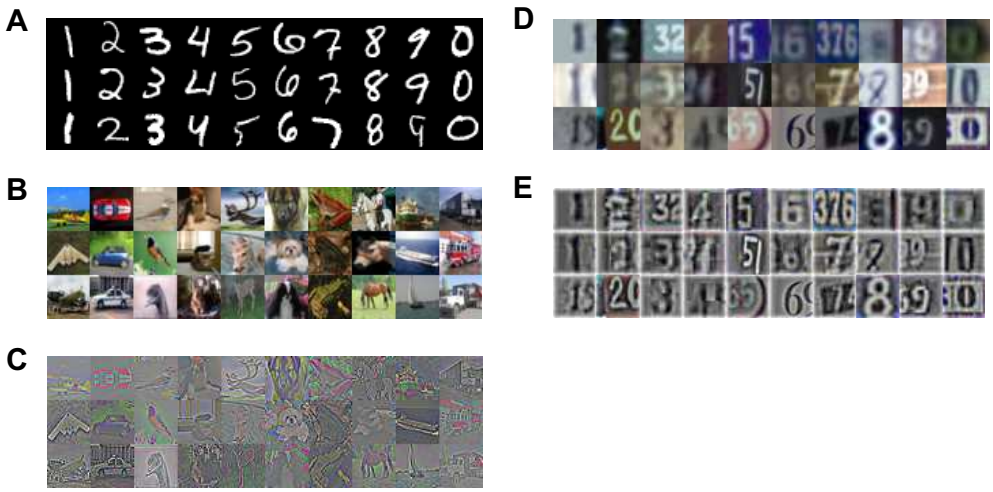

Figure 3.4: Three datasets we tested. (a) The MNIST. (b, c) The rawdata and its normalized version of the CIFAR-10. (d, e) The rawdata and its normalized version of the SVHN. The normalization is performed with local contrast normalization to manipulate the extreme brightness and color variations.

followed by 0.5 again for the third epochs. For example, the learning schedule 0.001 (100-20-10) denotes a learning rate of 0.001 for 100 epochs followed by 0.0001 for 20 epochs and 0.00005 for 10 epochs.

We evaluated the proposed regularization on three benchmark datasets: MNIST [57], CIFAR-10 [53], and SVHN [73], as depicted in Fig. 3.4. In the case of the CIFAR and SVHN, we preprocessed the data using zero-phase component analysis (ZCA) whitening and local contrast normalization; the same techniques as [31] and [118], respectively. These preprocessing techniques are known for normalizing the extreme brightness and color variations efficiently.

Our implementation was based on Caffe [47], which is one of the most popular deep learning frameworks. We added new forward-backward steps and customized four types of layers: convolutional, pooling, response normalization [40], and fully-connected layers. The codes and all the details of hyper-parameters [*i.e.*, weight decay and the number of hidden nodes] are available

on our GitHub page [1]. For each case, we presented a mean value with a standard deviation among 10 runs as a format of $\mu \pm \sigma$. For the generation of adversarial examples, we had to maintain an appropriate noise level β. This is shown in detail in the following subsection.

3.3.1 Improved classification performance

MNIST The MNIST [57] dataset is one of the most popular benchmarks and consists of a set of 28×28 grayscale images with corresponding labels 0–9. Because these images have high contrast like binary images, typically they have been tested without any pre-processing. To obtain desirable representations in the manifold space, we tested two types of models.

First, we trained models with two fully connected layers each with 800 hidden units, followed by a softmax layer. We used a learning schedule of 0.1 (40-40-20) and obtained a test accuracy of $98.848 \pm 0.052\%$, which is the best record over the published results with only two hidden layers. DropConnect [109] and dropout [97] produced $98.800 \pm 0.034\%$ and $98.720 \pm 0.040\%$, respectively.

Additionally, we conducted MRnet with two convolutional layers followed by two fully connected layers. The results are summarized in Table 3.1. The top four methods were performed 10 times using our implementation while the results of the bottom five methods were reported in the literature. The previous best published result is the 99.55% test accuracy of Maxout [31]. Among the alternatives compared, MRnet achieved the state-of-the-art discriminative performance: $99.536 \pm 0.045\%$ (best: 99.58%).

[1] https://github.com/taehoonlee/caffe

Table 3.1: Test set accuracy on the MNIST with convolutional hidden layers.

Method	Test accuracy (%)
MRnet + dropout	99.536 ± 0.045
Batch Normalization [44]	99.465 ± 0.051
Dropout [97]	99.482 ± 0.053
Only L2-decay	99.364 ± 0.055
DropConnect [109]	99.370 ± 0.035
Maxout + dropout [31]	99.55
NIN + dropout [63]	99.53
Stochastic Pooling [118]	99.53
Lasso in F-layers [46]	99.47

CIFAR-10 The CIFAR-10 [53] dataset is a set of 32×32 color images of 10 classes. There are 50,000 training and 10,000 test images. We applied ZCA preprocessing ($\epsilon = 0.01$) similar to [31]. To be consistent with previous work, we evaluated our method with 24×24 random cropping and horizontal flipping augmentation. The learning schedule was set to 0.001 (100-10-10).

The results of Caffe runs and the literature are summarized in Table 3.2. Note that a locally connected layer in the description column is a weight-unshared convolutional layer. We can achieve a test accuracy of $91.082 \pm 0.237\%$, which sets a new performance record.

SVHN The SVHN [73] dataset is composed of 604,388 training and 26,032 test images of 10 classes. Following [118], we conducted local contrast normalization with a 3×3 filter and 28×28 random cropping. The structure and parameters used in the SVHN are the same as those used for the CIFAR-10, which consists of three or four convolutional layers followed by two fully connected layers. For this dataset, we obtained a test accuracy of 97.521 ± 0.052

Table 3.2: Test set accuracy on the CIFAR-10 with data augmentation and the SVHN.

Method	Description	Test accuracy (%)	
		CIFAR-10	SVHN
MRnet + dropout	4C[1] + 2F[2]	91.182 ± 0.237	97.521 ± 0.052
MRnet + dropout	3C + 2F	89.835 ± 0.224	97.327 ± 0.048
Batch Normalization [44]	3C + 2F	89.032 ± 0.214	97.111 ± 0.047
Dropout [97]	3C + 2F	89.097 ± 0.240	97.058 ± 0.086
Only L2-decay	3C + 2F	88.259 ± 0.225	96.897 ± 0.058
DropConnect [109]	2C + 2L[3] + 1F	90.59	97.77 ± 0.039
Maxout + dropout [31]	3M[4] + 1F	90.62	97.53
NIN + dropout [63]	3N[5] + 1F	91.19	97.65
Stochastic Pooling [118]	3C	(no aug) 84.87	97.20

[1]a convolutional layer, [2]a fully connected layer, [3]a locally connected layer
[4]a maxout convolutional layer, [5]a network in network convolutional layer

with a learning schedule of 0.001 (20-20-20). A summary with the alternatives is provided in Table 3.2.

3.3.2 Disentanglement and generalization

The proposed manifold loss Φ minimizes the 2-norm difference between $a^{(L)}$ and $a'^{(L)}$. In other words, the difference in the last hidden layer's activations

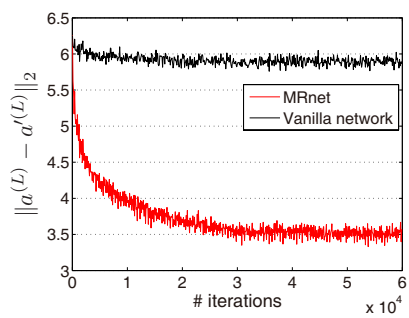

Figure 3.5: Difference between $a^{(L)}$ and $a'^{(L)}$ over iterations.

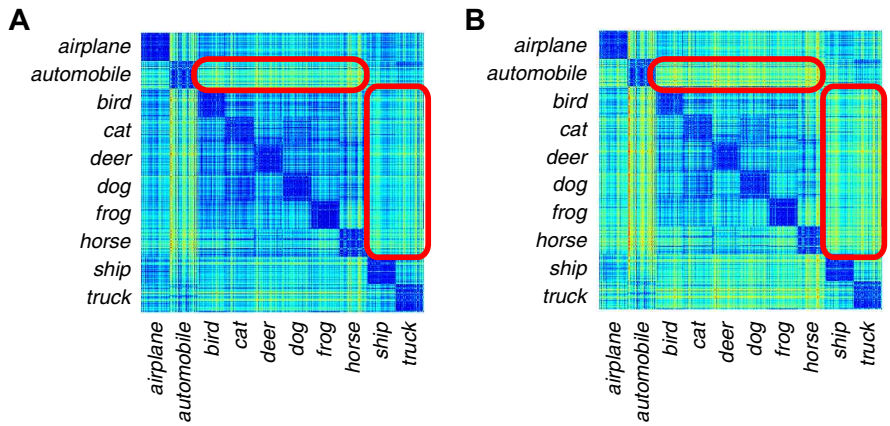

Figure 3.6: Embedding results of the CIFAR-10 test set. (a) Pairwise distance matrix of $a^{(L)}$ *without* Φ. (b) Pairwise distance matrix *with* Φ.

between an original and its adversarial sample is minimized. However, an adversarial perturbation at the beginning of training is meaningless. Thus, we applied Φ after training a vanilla neural network with several iterations, in a similar way to the pre-training phase of deep belief networks. Fig. 3.5 shows the variation of $\|a^{(L)} - a'^{(L)}\|_2$ over the number of iterations. The iteration 0 denotes the point at which Φ is applied.

As illustrated in Fig. 3.5, MRnet minimized the manifold distances more successfully than a vanilla network. The final value of $\|a^{(L)} - a'^{(L)}\|_2$ was 3.57. In the network we tested, the number of hidden nodes in the last hidden layer was 1024, and the average difference of individual activation values can be calculated approximately as 0.1 ($= \sqrt{3.57^2/1024}$). Because each activation value in the last hidden layer ranged from 0 to 2.74, the average difference 0.1 means that $a^{(L)}$'s and $a'^{(L)}$'s were projected into very closed regions by the proposed multi-layer manifold embedding.

We also visualized $a^{(L)}$ of the CIFAR-10 test set to examine the effect of the proposed manifold loss Φ. Fig. 3.6, Fig. 3.7, and Fig. 3.8 show the embedding

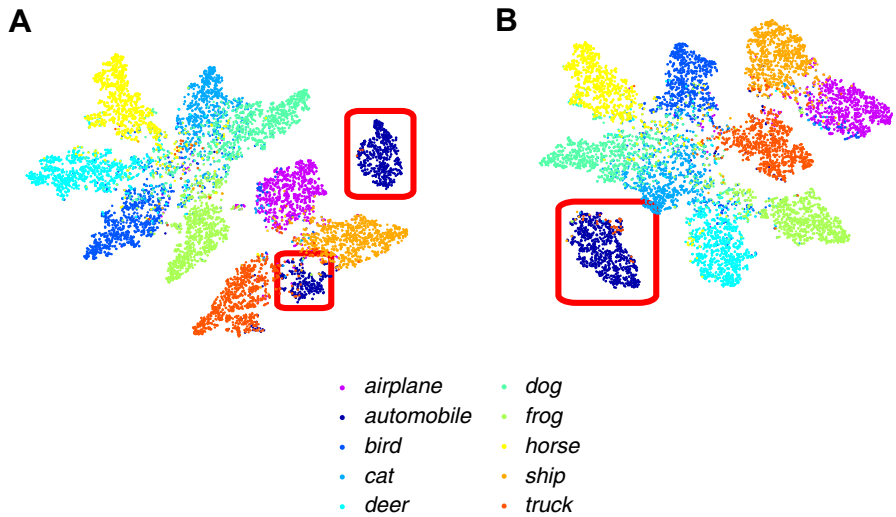

Figure 3.7: Embedding results of the CIFAR-10 test set. (a) 2-D visualization of the manifold embedding through t-SNE *without* Φ. (b) t-SNE plot *with* Φ.

results without and with Φ, respectively. As shown in Fig. 3.6(b), Φ increases the contrast of pairwise distances between intra-classes (block-diagonal elements) and inter-classes (other elements) compared with Fig. 3.6(a). The contrast can be quantified using two clustering evaluation metrics: the silhouette coefficients $\in [-1, 1]$ [86] and the Dunn index [22]. As clusters are separated better, both metrics produce higher values. Without and with Φ, the average values of silhouette coefficients do not exhibit significant differences. However, the Dunn indices without and with Φ were 0.0297 and 0.0433, respectively. Because the denominator of the Dunn index is the maximum distance within a cluster [22], we argue the Dunn index can present more appropriate quantification of the contrast, and obtained 31.45% improved contrast.

A similar effect can be found in the 2-D visualization with t-SNE [106] [see Fig. 3.7(a) and (b)]. Without Φ, two boxed chunks of the automobile class are separated and one is closed to the ship and truck classes. However, Φ can make instances of the automobile class be grouped into one chunk as depicted

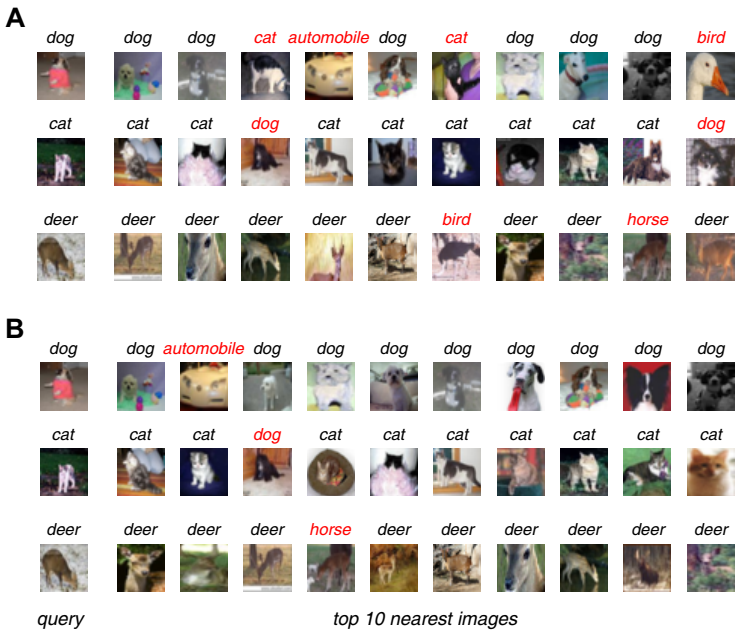

Figure 3.8: Embedding results of the CIFAR-10 test set. (a) Query images and top 10 nearest images *without* Φ. (b) query images with the nearest images *with* Φ.

in Fig. 3.7(b). Finally, Fig. 3.8 presents a few query images and the top 10 nearest neighbors on the manifold embedding space. The 10-th closest bird image of the dog and the 9-th closest bird image of the deer were eliminated in the list of nearest neighbors after applying Φ.

3.4 Summary

We have proposed a novel methodology, unifying deep learning and manifold learning, called manifold regularized networks (MRnet). Traditional neural networks, even state-of-the art deep networks, have intrinsic blind spots due to a huge number of parameters and linear function components using them. We tested MRnet and confirmed its improved generalization performance underpinned by the proposed manifold loss term on deep architectures. By exploiting

the characteristics of blind spots, the proposed MRnet can be extended to the discovery of true representations on manifolds in various learning tasks.

Chapter 4

Class-imbalance handling

4.1 Introduction

Fig. 4.1 shows an overview of the proposed methodology. As training data, we use a set of N DNA sequences, each of which has m nucleotides (nt). Some of these contain either an acceptor or a donor site, and the others contain no splice site. Each training sequence has a label: acceptor, donor, or non-site. Each sequence is converted into a binary vector by orthogonal encoding (see Section 4.1.1).

After preprocessing, our approach proceeds in two main steps: (1) unsupervised pre-training of component RBM using the proposed boosted contrastive divergence with categorical gradient; and (2) organizing DBN by RBM stacking and supervised fine-tuning of the DBN. The label of each training sequence is not used for the first pre-training step but only for the second fine-tuning step.

In the inference step (not shown in Fig. 4.1), an unlabeled DNA sequence is fed to the trained DBN, and it predicts whether the sequence contains a

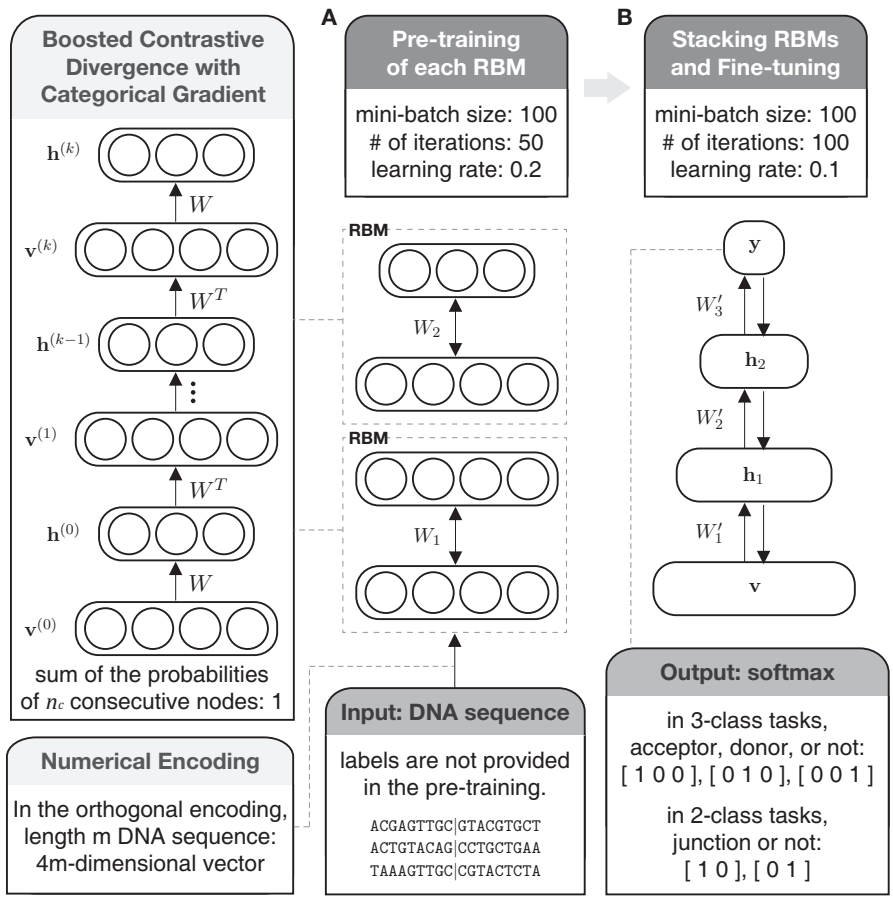

Figure 4.1: Proposed methodology: (a) pre-training; (b) fine tuning.

Input	A	G	T	T	G	C	T	G	
P	**0.22**	0.23	0.26	0.27	0.33	0.32	0.23	0.34	A
	0.31	0.20	0.25	0.30	0.21	**0.26**	0.30	0.21	C
	0.23	**0.27**	0.27	0.25	**0.23**	0.13	0.29	**0.32**	G
	0.24	0.30	**0.22**	**0.18**	0.23	0.29	**0.18**	0.13	T
Output	0.22	0.27	0.22	0.18	0.23	0.26	0.18	0.32	

Figure 4.2: An example of the frequency encoding ($k = 1$).

splice junction or not.

4.1.1 Numerical interpretation of DNA sequences

In case of the sequence alignment based methods, we do not need to obtain numerical interpretation of sequences because the sequences are mapped to the reference in themselves. However, machine learning based approaches require pre-processing of categorical features (*i.e.*, A, C, G, T of DNA sequences). In this section, we describe how DNA sequences are converted into numerical vectors and how deep learning architecture works.

The four types of nucleotides organizing DNA are adenine (A), guanine (G), cytosine (C), and thymine (T). We can acquire the genetic information embedded within DNA as alphabet sequences (*i.e.*, ...AGTTGCTG...). Before enumeration of the encoding schemes, we define k-mer as a subsequence generated by a moving window of size k in the sequence. For example, AGTTGCTG can generate a set of 2-mers AG, GT, TT, TG, GC, CT, TG. That is, the term k-mer is identical to n-gram in the field of computational linguistics, and all the schemes described in this paper can be coupled with the k-mer concept.

As shown in Table 4.1, we categorize the encoding schemes into two types: location-based and feature-based. The location-based method refers to a transformation preserving the spatial information. For instance, AGTTGCTG can be

Table 4.1: The encoding schemes

Number Type	Name	Rules
		Location-based Type
Binary	Simple 2-bit	A − 00, C − 01, G − 10, T − 11
	Orthogonal [3]	A − 1000, C − 0100, G − 0010, T − 0001 (monomer) AA...A − 100..., AA...C − 010..., ⋯, TT...T − ...001 (k-mer)
	Physicochemical [33]	A, C (amino) − 1, G, T (keto) − 0 (functional group) A, G (purine) − 1, C, T (pyrimidine) − 0 (ring structure) C, G (strong-H) − 1, A, T (weak-H) − 0 (hydrogen bond)
	Parity [66]	A − 1010, C − 1001, G − 0110, T − 0101 (physicochemical + a parity bit)
Integer	MN, PN [43]	A − 1, C − 2, G − 3, T − 4 (MN, mono-nucleotide) AA − 1, AC − 2, AG − 3, ⋯, TG − 15, TT − 16 (PN, pair-wise nucleotides)
Frequency	Frequency	1. Construction of a matrix P where p_{ij} is a frequency of the i-th letter (or k-mer) in the j-th position over all the sequences. 2. Translation into a frequency vector using P (refers to Fig. 4.2).
	Bayes kernel [119]	After creation of $P^{(p)}$ and $P^{(n)}$ over all the positive and negative samples, translation into a frequency vector $x = [x_{P^{(p)}}\ x_{P^{(n)}}]$.
	FDTF [43]	Using $P^{(p)}$ and $P^{(n)}$, translation into a frequency vector $x = x_{P^{(p)} - P^{(n)}}$.
		Feature-based Type
Frequency	k-mer counting [67]	Translation into a 4^k-dimensional vector $x = [x_1, \cdots, x_{4^k}]$ where x_i's are the number of occurrences of AA...A, AA...C, ⋯, TT...T.
Entropy	Codon-entropy [112]	Translation into a 4^3-dimensional vector $x = [x_1, \cdots, x_{4^3}]$, (each dimension measures the periodicity of i-th codon (3-mer).)

translated into a vector $x = [1\ 3\ 4\ 4\ 3\ 2\ 4\ 3]$ where a number in each dimension corresponds to a letter in each position. The feature-based method extracts certain features with losing some information of sequences. The details of each type are described as follows.

In the location-based schemes, a letter (monomer) or a k-mer in each position can be represented as binary, integer, or frequency. The simple 2-bit creates a set $\{00, 01, 10, 11\}$ for the four monomer, and the orthogonal encoding does more sparse codes in which each k-mer has no algebraic correlations between the other k-mers [3]. Besides, we can convert each k-mer pattern into an integer by using MN (mono-nucleotide) or PN (pair-wise nucleotides) encoding [43]. Physicochemical encoding [33] means that each bit is an indicator for physicochemical characteristics such as amino/keto, purine/pyrimidine, and strong-H/weak-H. The authors of [66] proposed the addition of a parity bit to the characteristics.

The frequency methods measure frequencies of the k-mers with the moving window of size k. We illustrate the frequency method with an example ($k = 1$) as shown in Fig. 4.2. First, we calculate a matrix P where p_{ij} is a frequency of the i-th letter in the j-th position over all the given sequences. By definition, $p_{32} = 0.27$ indicates that 27 out of 100 sequences have a letter '`G`' in the second position. Using the matrix P, it is straightforward to understand that `AGTTGCTG` becomes a vector, $x_P = [0.22\ 0.27\ 0.22\ 0.18\ 0.23\ 0.26\ 0.18\ 0.32]$. The Bayes kernel [119] and the FDTF (Frequency Difference between the True sites and False sites) [43] are based on the frequency approach. These methods construct two different matrices $P^{(p)}$ and $P^{(n)}$ using all the positive (true splice site) and negative samples. Eventually, the Bayes kernel yields a frequency vector $x = [x_{P^{(p)}}\ x_{P^{(n)}}]$ and the FDTF returns a vector $x = x_{P'}$

where $P' = P^{(p)} - P^{(n)}$.

Consequently, the location-based schemes have their own inverse functions because they are one-to-one correspondences between a set of sequences and a set of vectors. On the other hand, the feature-based schemes may not have inverses. The k-mer counting [67] (known as n-gram model in the machine learning field) counts the number of occurrences of all the possible subsequences. Namely, `AGTTGCTG` produces a vector $x = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 2\ 1]$ where individual dimensions are frequencies of the length $k = 2$ substrings `AA`, `AC`, `AG`, $\cdots$, `TG`, `TT`. Furthermore, we can replace a frequency with a entropy. The transformation proposed in [112] measures the periodicity of occurrences of codons (3-mers in DNA or RNA).

In summary, feature descriptors based on k-mer counting or entropy are not recoverable to its original sequence. However, they have two advantages over the location-based descriptors; the feature-based encodings can generate short codes for efficient computation and robust codes for indel errors although losses of the spatial information arise. The location-based encodings preserves the information and is vulnerable to indel errors at the same time. Also, it makes long codes which may lead to the curse of dimensionality and limited scalability as a sequence goes longer.

In this study, we employ n_c-bit 1-hot encoding because the orthogonal encoding (*e.g.,* 1-hot encoding) is widely used [3] for biological sequences. For $n_c = 4$, `A`, `C`, `G`, and `T` are encoded by $1000, 0100, 0010$, and 0001, respectively (we use the notations 1000 and $[1, 0, 0, 0]$ interchangeably).

Orthogonal encoding may cause the trained model to have limited generalization ability because of the sparsity of encoding. For example, sequence `AGTT` is encoded by 16-dimensional binary vector 1000001000010001, of which

75% of the elements are zero. To alleviate this issue, we devise a new regularization technique that incorporates prior knowledge on the sparsity, as will be detailed shortly.

4.1.2 Review of junction prediction problem

In living organisms, biological information flows from deoxyribonucleic acid (DNA) to ribonucleic acid (RNA) to protein. DNA is a sequence of four types of nucleotides: adenine (`A`), guanine (`G`), cytosine (`C`), and thymine (`T`). A gene is a segment of DNA that constitutes the basic unit of heredity. As shown in Fig. 4.3, genetic information is delivered from DNA to protein through a procedure called *gene expression* [64]. There are three major steps in gene expression: transcription, splicing, and translation.

Eukaryotic genes have an internal structure that includes two types of subunits: exons (protein-coding regions) and introns (non-coding regions). Introns intervene between exons, and the boundary between an exon and an intron is referred to as the *splice junction (site)*. The majority of splice sites contain consensus strings called *canonical splicing* patterns. The most frequent patterns are dimer `GT` (called *donor*) and dimer `AG` (called *acceptor*) at intron/exon boundaries [12].

During transcription, DNA is copied into precursor messenger RNA (pre-mRNA), and then the introns in the transcribed pre-mRNA are removed by splicing [49]. For a single gene, various combinations of alternative exons selectively remain in the resulting mature mRNA, allowing the construction of multiple proteins from the gene. Consequently, this alternative splicing gives rise to the enormous diversity of proteins [75], and the identification of splice sites is a crucial step to fully understand gene expression.

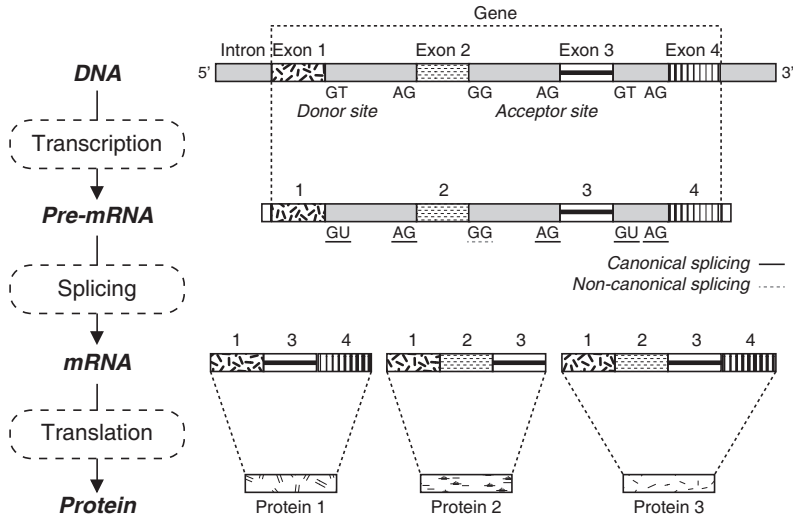

Figure 4.3: Gene expression process. Dimers `GU`(`GT`) and `AG` represent canonical donor and acceptor splice sites, respectively. Dimer `GG` shows an example of non-canonical (donor) sites.

Existing junction prediction methods belong two categories: sequence alignment-based and machine learning-based. Alignment-based strategies [105, 110, 32] reconstruct exons by mapping millions of short RNA sequences to the whole genome sequence and then estimate where splicing occurs using adjacent exons. Despite the need for a reference genome, alignment methods can identify novel splice sites in addition to the original sequence locations in the reference. Alternatively, machine-learning techniques construct a descriptive model of splicing by training with known junction signals. The learning models used include artificial neural network (ANN, [98, 77, 11]), support vector machine (SVM, [21, 43, 96]), and hidden Markov model (HMM, [83, 81, 4]).

Learning-based approaches have produced promising results since the early 1980's, but they often suffer from practical limitations, such as excessive false positives and limited scalability. With the advent of next-generation sequencing technology, alignment-based methodologies using RNA-seq technology [17]

Algorithm 1 Boosted CD with Categorical Gradient

Input: N encoded DNA sequences $\mathbf{v}_1, \ldots, \mathbf{v}_N$
Output: weights $W, \mathbf{b}, \mathbf{c}$
Initialize $W \sim \mathcal{N}(0, 0.1), \mathbf{b} = \mathbf{0}, \mathbf{c} = \mathbf{0}$
for each epoch **do**
  for each minibatch with size N **do**
    Compute $E_{min} = -\sum_i b_i - \sum_j c_j - \sum_i \sum_j w_{ij}$
    for $n = 1$ to N **do**
      Compute $\mathbf{h}_n^{(0)} = P(\mathbf{h} = 1|\mathbf{v}_n^{(0)})$
      Sample $\mathbf{v}_n^{(1)}$ from $P(\mathbf{v} = 1|\mathbf{h}_n^{(0)})$
      Compute $\mathbf{h}_n^{(1)} = P(\mathbf{h} = 1|\mathbf{v}_n^{(1)})$
      Compute $\alpha_n = E(\mathbf{v}_n^{(1)}, \mathbf{h}_n^{(1)}) - E_{min}$
    end for
    Normalize $\alpha_n = N \cdot \alpha_n / \sum_n \alpha_n$ for each n
    Update W, $\mathbf{b}$, $\mathbf{c}$ using (4.1), (4.2), (4.3) with α_n's
  end for
end for

have gained popularity over the last decade as an alternative to learning-based approaches. However, existing alignment-based methods, such as TopHat [104] and SpliceMap [2], consider only *canonical* splicing signals (`GT` or `AG`) in their splitting and merging procedures and often miss important splicing signals. To improve the accuracy of prediction, not only canonical but also *non-canonical* patterns are important. Given that machine learning-based methodologies can learn and predict such non-canonical junction signals, we believe that learning-based and alignment-based approaches should be used in a complementary way to boost performance.

4.2 Methods

4.2.1 Training RBM using boosted contrastive divergence with categorical gradients

For a DNA sequence of m nucleotides, we convert it into a binary vector $\mathbf{v}$ of $n_v = (n_c \times m)$ elements by the n_c-bit 1-hot encoding as described previously. This $\mathbf{v}$ becomes the visible units of the RBM described in Section 2.3.1. In Eq. 2.2, $\mathbf{v}^{(0)}$ and $\mathbf{v}^{(k)}$ can be considered as the original input and its reconstruction vectors, respectively. We require that the sum of the probabilities of n_c consecutive nodes in the reconstructed $\mathbf{v}^{(k)}$ units be 1. We add a regularization term that penalizes the deviation of the sum of n_c visible units from 1. We called this term the *categorical gradient*.

The NLL of input data now becomes

$$\min_{W,\mathbf{b},\mathbf{c}} \mathbf{E}\left[-\sum_{n=1}^{N}\log P(\mathbf{v}_n)\right] + \frac{\lambda_c}{2}\sum_{i=1}^{m}(\sum_{j=1}^{n_c} v^{(k)}_{n_c(i-1)+j} - 1)^2$$

where N is the number of input sequences, and λ_c represent sensitivity to the categorical gradient. The update rules for minimizing the aforementioned NLL are as follows:

$$\frac{\partial L}{\partial W} \approx \text{Eq. (2.2)} + \frac{1}{N}\sum_{n=1}^{N} f(\mathbf{v}_n^{(k)})\mathbf{h}_n^{(k-1)} \tag{4.1}$$

$$\frac{\partial L}{\partial \mathbf{b}} \approx \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{v}_n^{(0)} - \mathbf{v}_n^{(k)} + f(\mathbf{v}_n^{(k)})\right) \tag{4.2}$$

$$\frac{\partial L}{\partial \mathbf{c}} \approx \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{h}_n^{(0)} - \mathbf{h}_n^{(k)}\right) \tag{4.3}$$

$$f(\mathbf{v}) = \mathbf{v}\circ(1-\mathbf{v})\circ g(\mathbf{v}),\ g(\mathbf{v})_i = \sum_{j=1}^{n_c} v_{n_c[\frac{i-1}{n_c}]+j} - 1$$

where operator $\circ$ denotes the Hadamard product and $g(\mathbf{v})$ represents an element-wise operation that replicates the deviations of the sum of the n_c nodes from 1. For example, $g(\mathbf{v}) = [g_1, g_1, g_1, g_1, g_2, g_2, g_2, g_2]$, where $g_1 = \sum_{i=1}^{4} v_i - 1$ and $g_2 = \sum_{i=5}^{8} v_i - 1$, for $n_v = 8$ and $n_c = 4$.

Using these modified derivatives allows us to extract novel features that would work well for both reconstruction and classification. For training with the proposed categorical gradient, we propose a new approach that improves the CD-1 approach. Conventional CD often provides a reasonable approximation of the model distribution but still suffers from a critical drawback: the computation of the negative phase needs to be more precise than $\mathbf{v}^{(k)}\mathbf{h}^{(k)^T}$ to approximate the model expectation $\mathbf{E}_{model}[\mathbf{v}\mathbf{h}^T]$. There have been approaches to address the issue, such as persistent CD [102] and parallel tempering [15]. They often draw samples from the model distribution more accurately than conventional CD, but their approximations can still be far from the model expectation.

If we assign the same weight to all the data, the performance of Gibbs sampling would degrade in the regions that are hardly observed. To approximate the model expectation precisely, we need to sample these regions. Borrowing the idea of boosting in ensemble learning, we emphasize unstable observations that have high energy during the training procedure, given that high-energy states typically have low likelihood and provide a high reconstruction error. When sampling, we therefore weight each observation by the energy of its reconstruction $E(\mathbf{v}_n^{(k)}, \mathbf{h}_n^{(k)})$. This re-weighting is also linked to importance sampling, in which the density function is scaled in order to move the probability mass to the desired event region [72].

The rationale behind our idea is as follows: At the beginning of training,

the joint probability distribution $P(\mathbf{v}, \mathbf{h})$ is highly unstable and the update direction is affected by the first several mini-batches. If important observations are not included therein, the possibility of sampling further from these regions decreases because RBM assigns high energy to these regions. CD training is looped over all mini-batches and can alleviate this issue to some extent. However, when there is a significant class imbalance, as in the junction prediction, we are not likely to extract appropriate hidden representations of those observations.

Algorithm 1 presents the pseudocode of our training algorithm, which is named *boosted CD with categorical gradients* to emphasize the notion of re-weighting. We first compute the minimum energy E_{min} under the current configuration $(W, \mathbf{b}, \mathbf{c})$ and assign energy-proportional weight α_n to individual data $\mathbf{v}_n^{(0)}$. We then normalize these α's so that the coefficients vary from 0 to N. Most of the coefficients will be around 1 because most of the energy values deviate from E_{min}. Combining α_n's with the update rules is straightforward. For example, Eq. 4.1 becomes $\frac{1}{N}\sum_{n=1}^{N}\alpha_n\left(\mathbf{v}_n^{(0)}\mathbf{h}_n^{(0)^T} - \mathbf{v}_n^{(k)}\mathbf{h}_n^{(k)^T} + f(\mathbf{v}_n^{(k)})\mathbf{h}_n^{(k-1)}\right)$.

4.2.2 Stacking and fine-tuning

After pre-training RBMs with the proposed approach, we stack them and place an output layer on them to construct a DBN (see Fig. 4.1) and then train it in a supervised manner.

For three-class problems, the output softmax layer consists of three nodes and the resulting output vector $\mathbf{y}$ is one of the following: 100, 010, or 001 for acceptor, donor, and non-site, respectively. For two-class problems, the output layer consists of two nodes, and $\mathbf{y}$ is either 10 or 01 for splice-site (donor/acceptor) or non-site, respectively.

In the fine-tuning step, we utilize backpropagation to minimize the squared loss function $J(W, \mathbf{c}) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{2}||f(\mathbf{v}_n) - y_n||_2^2$, where $f(\mathbf{v}_n)$ and y_n are the final output vector of the network for input $\mathbf{v}_n$ and the desired output vector, respectively. Through this optimization, pre-trained W_1, W_2, and W_3 are fine-tuned to the final weights W_1', W_2', and W_3', respectively.

4.2.3 Initialization and parameter setting

To speed up the pre-training and fine-tuning procedures, the input datasets were split into mini-batches of $M = 100$ sequences.

In the pre-training step, we initialized the values of W in component RBMs using Gaussian $\mathcal{N}(0, 0.1)$ and set both $\mathbf{b}$ and $\mathbf{c}$ to the zero vector. We set the number of iterations and the learning rate to $T = 50$ and $\alpha = 0.2$, respectively. We used a different number of hidden nodes or layers for different datasets, as will be explained in the next section.

In the fine-tuning stage, the weights (W_3, $\mathbf{c}_3$) for the output layer were initialized using the Gaussian $\mathcal{N}(0, 0.1)$ and the zero vector, respectively. The number of iterations and the learning rate used were $T = 100$ and $\alpha = 0.1$.

4.3 Results and discussion

4.3.1 Experiment preparation

We tested our approach with the datasets listed in Tables 4.2 and 4.3. The genome-wide datasets for humans (GWH, [96]) consist of two types of datasets: GWH-donor and GWH-acceptor. Each of these two datasets includes sequences from the 24 human chromosomes (22 autosomes and 2 sex chromosomes). All the sequences in both GWH-donor and GWH-acceptor are of length 398nt.

Table 4.2: GWH genome-wide data [96]

two-class, 398nt long, contains canonical signals only		
Data ID	# of positives	# of negatives
GWH-donor	160,601 (0.21%)	76,335,126
WH-acceptor	158,217 (0.29%)	54,469,623

Table 4.3: UCSC genome browser database [48]

three-class, 60nt long, contains non-canonical signals as well			
Data ID	# of donors	# of acceptors	# of non-site
UCSC-hg19	62,819	62,819	62,819
UCSC-hg38	63,454	63,454	63,454

The splice signals from these sequences are all canonical, and all the sequences have dimer `GT` or `AG` in the middle. That is, each sequence from GWH-donor has dimer `GT` in nucleotide positions 200 and 201, and each sequence from GWH-acceptor has dimer `AG` in positions 198 and 199. These dimers indicate true splice sites for positive examples, whereas they do not represent splice sites for negative examples.

For the GWH data, there is a substantial imbalance between the numbers of positive and negative examples: only 0.21% (0.29%) of the examples are positive for GWH-donor(acceptor). To see the effect of having this imbalance in training, we randomly sampled negative examples and used them, thus varying the so-called *decoy rate*[1] [96] from 5 to 15.

While the GWH datasets are for two-class classification (splice sites or not), the UCSC datasets [48] are for three-class classification (donor, acceptor, or neither). The UCSC-hg38 dataset contains 24,279 genes with 1–173 (on

[1]Denoted by $r = \#$ negative samples/# positive samples

average 9.44) exons per gene. The UCSC datasets also consist of the sequences from 24 chromosomes. Most of these exons have duplicates in the annotation database due to alternative splicing [49]. We randomly chose 63,454 unique exons out of 229,255. According to [77], we then generated three examples by taking the sequences centered at the left, middle, and right boundaries of each exon. They correspond to acceptor, non-site, and donor examples, respectively. The UCSC-hg19 dataset was also utilized to generate additional examples in the same manner. Both UCSC datasets include non-canonical splice signals (*i.e.*, other than `GT` and `AG`) in addition to canonical signals.

We carried out all the experiments using MATLAB. For comparison with SVM, we used the LIBSVM package [13]. Deepmat code [15] was used for the implementation of persistent CD and parallel tempering.

In the following, the architecture of a DBN is denoted by the number of nodes in each layer. For instance, a 1592-160-16-2 DBN has four layers with 1592 input units, 160 units in the first hidden layer, and so on.

4.3.2 Improved prediction performance and runtime

To evaluate the prediction performance of our approach, we measured the F1-score and accuracy values[2], and the runtime, as shown in Fig. 4.4 and Fig. 4.5. For comparison, we also included SVM (with RBF and sigmoid kernels) and two existing tools for splice junction prediction: GeneSplicer [81] and SpliceMachine [21], which are based on decision trees and linear SVM, respectively. Note that these two existing tools were designed only for two-class problems and we were able to test them for the GWH data only. We used (398×4)-160-16-2 DBN for the GWH data and (60×4)-120-30-3 DBN for the

[2] F1-score $= 2TP/(2TP + FP + FN)$, accuracy $= (TP + TN)/(P + N)$

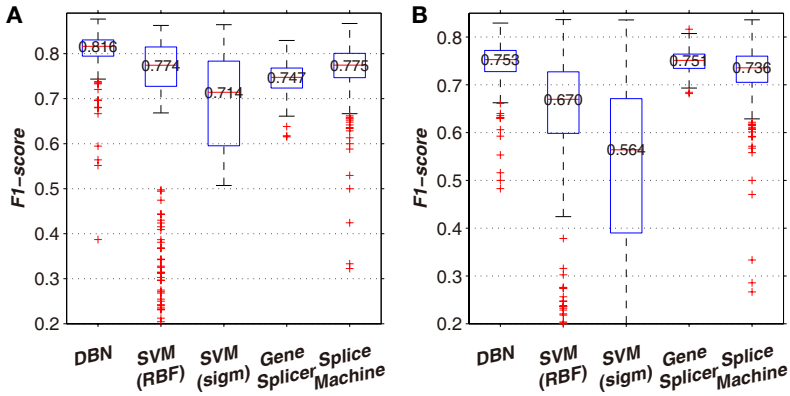

Figure 4.4: Comparison of classification performance: (a) F1-score for GWH-donor; (b) F1-score for GWH-acceptor.

UCSC data.

As shown in Fig. 4.4(a) and Fig. 4.5(b), the proposed method outperformed the existing state-of-the-art prediction tools in terms of the F1-score and accuracy. Note that we performed 10-fold cross validation for each of the 24 individual chromosomes to test a single dataset and each of the boxplots shows the distribution of 240 values. Quantitatively, our method showed 4.1–10.2% and 0.2–18.9% higher performance in terms of the median F1-score for the GWH-donor and GWH-acceptor datasets, respectively. For the UCSC datasets, our method produced 2.0–2.4% and 2.1–2.7% higher median accuracy than the SVM method.

Fig. 4.5(c) shows the runtime of the three different methods measured on the chromosome 1 dataset (the largest one) in UCSC-hg38. The proposed approach ran 3.86 times faster and 4.15 times faster than the SVM with the RBF and sigmoid kernel functions, respectively, in the worst case. Although the computational complexity of DBN depends on the numbers of layers and nodes, we still expect that DBNs will run faster than SVM for large datasets. This is because SVMs require quadratic programming, which takes $O(N^2)$

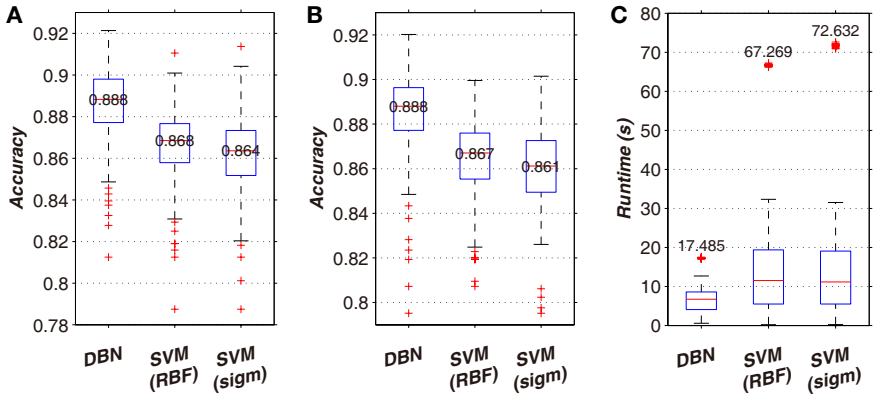

Figure 4.5: Comparison of classification performance: (a) accuracy for UCSC-hg19; (b) accuracy for UCSC-hg38; (c) runtime for UCSC-hg38 (chromosome 1).

time with standard interior-point methods [1]. On the other hand, DBN takes only $O(N)$ because the number of computations for updating gradients is proportional to N.

4.3.3 More robust prediction by proposed approach

We further tested our approach in terms of robustness to the input sequence and imbalance in training examples, as shown in Fig. 4.6. The dataset used came from the chromosome 20 part in the GWH-acceptor data.

Fig. 4.6 (a1–a3) shows how the performance measures[3] (F1-score, precision, and recall) change as we vary the sequence length m from 38 to 398 with decoy-rate r fixed at 5. The DBN architecture was accordingly changed from (38×4)-160-16-2 to (398×4)-160-16-2. With increasing m, the SVM method produced slowly increasing precision, while rapidly decreasing recall and the F1-score. This indicates that longer sequences incur a rapid increase in the number of false negatives for SVM. Thus, SVM is apt to miss true splice sites

[3]Precision $= TP/(TP + FP)$, recall $= TP/(TP + FN)$

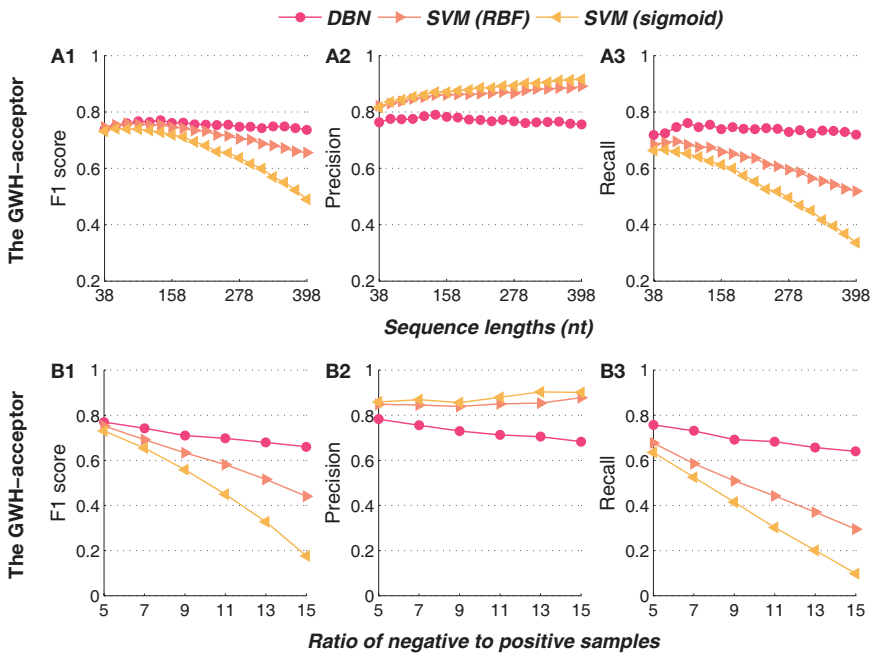

Figure 4.6: Effects of sequence length m and decoy rate r on performance: (a) varying m from 38 to 398 with fixed $r = 5$; (b) varying r from 5 to 15 with fixed $m = 138$. [data: chromosome 20 in GWH-acceptor]

if too many bases are considered around a `GT` or `AG` dimer. Longer sequences also resulted in reduced F1-scores for DBN, but the degree of reduction was noticeably smaller for our DBN method, which indicates a higher level of robustness to sequence lengths. DBN achieved the best F1-score at 77.13% with length $m = 138$.

To see the effect of decoy-rate r, we varied r from 5 to 15 for $m = 138$ with (138×4)-160-16-2 DBN and measured the F1-score, precision, and recall, as shown in Fig. 4.6 (b1–b3). Similarly to the experiments on sequence lengths, the proposed DBN approach outperformed the alternative in terms of r, suggesting that our approach can cope with the imbalance in training samples better. For instance, as r increased, so did the precision of SVM, simply because it predicted the label of most examples to be negative. The other

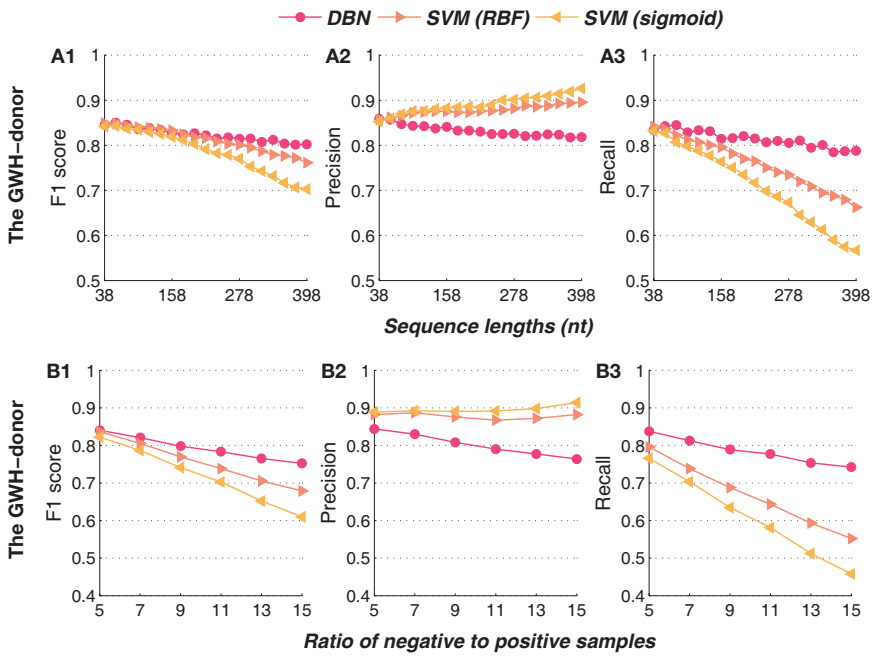

Figure 4.7: Effects of sequence length m and decoy rate r on performance: (a) varying m from 38 to 398 with fixed $r = 5$; (b) varying r from 5 to 15 with fixed $m = 138$. [data: chromosome 20 in GWH-donor]

measures (recall and F1-score) of SVM decreased more significantly than those of DBN as we increased r.

4.3.4 Effects of regularization on performance

There exist conventional RBM-based approaches to modeling categorical data by normalizing the probabilities of binary units in a softmax way [90]. By contrast, our approach utilizes a regularization term for applying RBM to model discrete data, allowing the sum of probabilities to slightly deviate from 1 if that is helpful for minimizing energy.

For a performance comparison, we measured the training and test error of three types of RBMs (basic, softmax, and regularized) using samples of the chromosome 19 sequences in the GWH-donor data. In the results shown in

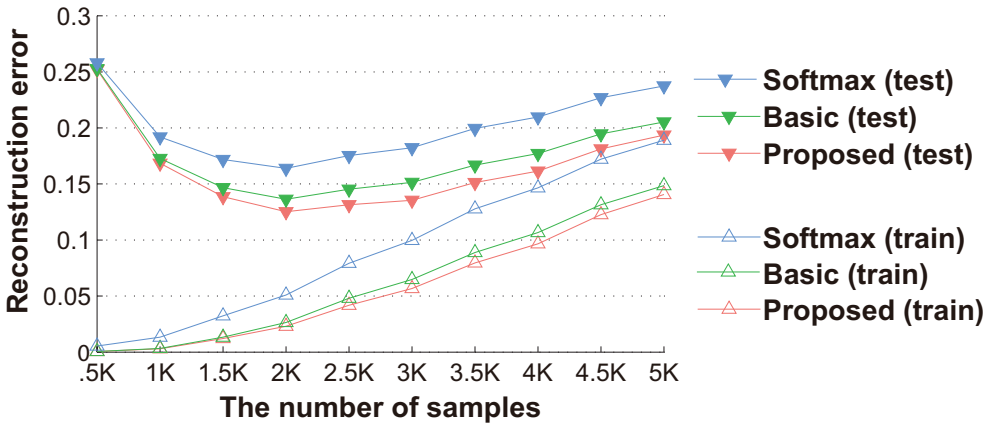

Figure 4.8: Comparing three types of RBMs (basic, softmax, and proposed) in terms of the reconstruction error. [data: chromosome 19 in GWH-donor, sequence length: 200nt, # iterations: 500, learning rate: 0.1, L2-decay: 10^{-3}, λ_c: 0.02]

Fig. 4.8, the proposed regularization-based RBM consistently outperformed the softmax version in training and test error. When the number of samples is less than 2,000, we observe overfitting (*i.e.*, decreasing training error with increasing test error) for all the approaches compared.

A part of the motivation for our approach comes from the tradeoff between minimizing energy and maintaining the probability sum at 1. Given that having low energy is likely to produce low reconstruction error, the proposed regularized RBM succeeded in achieving lower error by slightly sacrificing the probability sum constraint. Leveraged by the regularization term, our approach could find more appropriate hidden representations than the alternatives.

4.3.5 Efficient RBM training by boosted CD

To further validate the proposed boosted CD training, we tested it with a modified version of the MNIST dataset [57]. To simulate a class-imbalance

Table 4.4: Test accuracy by different training methods

Method ↓ # of samples →	12,000	25,000	60,000
Boosted CD (proposed)	96.09%	95.69%	98.23%
CD	94.49%	94.36%	98.17%
Persistent CD	45.58%	46.46%	98.36%
Parallel tempering†	95.84%	95.74%	98.52%

†approximately 10 times slower than boosted CD

situation, we randomly dropped observations with different drop rates for different classes and created two training sets (with 12,000 and 25,000 samples each) and a test set (with 10,000 samples). We repeated this procedure 10 times.

Table 4.4 lists the classification accuracy (averaged over the 10 runs) obtained by a 784-200-100-10 DBN trained with four different methods using the test data. As reference, the table also shows the accuracy values from the unmodified MNIST dataset (with 60,000 training examples).

For the two class-imbalance cases, proposed boosted CD and parallel tempering showed the best performance. However, due to the need for extra sampling, the time demand of parallel tempering was approximately 10 times higher to achieve the level of accuracy comparable to boosted CD. The performance of persistent CD was notably deteriorated for the class-imbalance cases. In such cases, the Gibbs sampler in training would hardly draw samples from low-density (but important) regions. Because training by persistent CD continues sampling from the Gibbs chain of the previous iterations, errors may have accumulated, giving unsatisfactory performance.

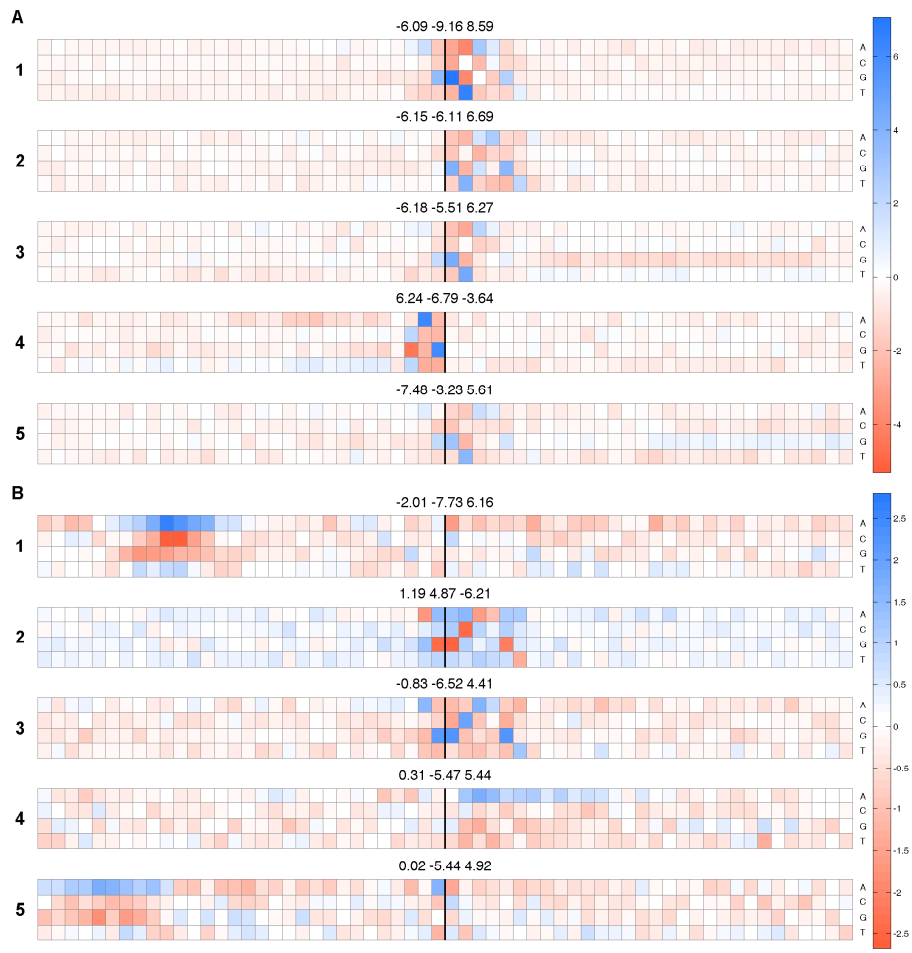

Figure 4.9: Top discriminative features for UCSC datasets: (a) including canonical splicing sites; (b) excluding canonical splicing sites.

4.3.6 Identification of non-canonical splice sites

To recognize non-canonical splice sites, such as GC pairs at donor sites, we trained the models using a combination of the two UCSC datasets. The merged dataset contained 378,819 sequences and the numbers of donors with GT pairs and acceptors with AG pairs were 109,000 (86.32% of donors) and 106,868 (84.63% of acceptors), respectively. First, we trained the 240-120-3 DBN with both the UCSC-hg19 and UCSC-hg38 datasets, including canonical splicing. The other parameters were the same as in the previous experiments. The training yielded two weight matrices, $W_1 \in \mathbf{R}^{240\times120}$ and $W_2 \in \mathbf{R}^{120\times3}$. Here, we can regard the two matrices as a set of 120 feature vectors and the discriminative scores for the three classes, respectively. In other words, each column in W_1 is a feature vector and its corresponding row in W_2 is composed of three weight values for the three classes.

The discriminability of a feature vector can be defined by the variance of the corresponding row in W_2. That is, a feature vector is not capable of discriminating any classes when the differences among the three discriminative scores of the feature vector are close to zero. The feature vectors were ranked in order of discriminability and the five most discriminative patterns are shown in Fig. 4.9(a). The vectors were reshaped into matrix forms (each row denotes A, C, G, and T) and colored according to weight values. A darker blue represents a higher positive value, whereas a darker red represents a lower negative value. For example, we can infer the most likely sequence from Fig. 4.9(a1) as '..GT..' because the third and fourth rows at the boundary are bold blue. The three numbers presented above each template represent the three discriminative scores for acceptor, non-boundary, and donor, from left to right.

As expected, we observed that the most discriminative features of Fig. 4.9(a)

```
1        NNNNNNNNAAAAAANNNNNNNNNNNNNNNN|NNNNNNNNNNNNNNNNNNNN
2        NNNNNNNNNNNNNNNNNNNNNNNNNNNNAG|GCAAGTNNNNNNNNNNNNNN
3        NNNNNNNNNNNNNNNNNNNNNNNNNNNNAG|GCANGTNNNNNNNNNNNNNN
4        NNNNNNNNNNNNNNNNNNNNNNNNNNNNNN|NAAAAAANANNNNNNNNNNN
5        NNAAAAANANNNNNNNNNNNNNNNNNNNNA|NNNNNNNNNNNNNNNNNNNN
```

Figure 4.10: The most likely sequences representing non-canonical splice sites inferred from analyzing Fig. 4.9(b).

are '..GT..' (a1, a2, a3, and a5) and '..AG..' (a4). Because only a few examples of non-canonical splicing were included, it was hard for the feature vector to detect the subtle signals. Therefore, we trained the same DBN again, using the 162,951 examples that remained after excluding the canonical splice sites. Fig. 4.9(b) shows the five best patterns for the non-canonical sites. We can derive the most likely sequences from the best weight vectors, as seen in Fig. 4.10. We found that non-canonical splicing arose when introns contained GCA or NAA sequences at their boundaries or contiguous A's occurred around the boundaries in exon regions. Using these methods, we may be able to reveal more novel patterns related to exons and alternative splicing that otherwise cannot be identified using existing machine learning techniques.

4.4 Summary

We have presented a novel DBN-based approach to splice site prediction at DNA level. Our contributions include the following:

- A new RBM training method called *boosted CD with categorical gradients* that improves conventional CD;
- Significant boosts in splice site prediction in terms of accuracy and runtime, along with reduced susceptibility to sequence lengths;

- Increased robustness when handling high-dimensional class-imbalanced data;
- The ability to detect subtle non-canonical splicing signals that often could not be identified by traditional methods.

Given the accuracy, efficiency, and robustness of our DBN-based methodology, we anticipate that it can be extended to the discovery of primary structural patterns of other genomic elements that are often too subtle to detect using existing computational techniques.

Chapter 5

Insufficient data handling

5.1 Introduction

DNA damage is known to be a major cause of cancer and many aging-related diseases [42]. The comet assay, also known as the single cell gel electrophoresis, can directly visualize DNA damage in eukaryotic cells [24]. After lysing of DNA and applying of an electric field, the extent of DNA strand breaks can be determined based on the resulting images in which individual cells appear as 'comets' (see Fig. 5.1). Since developed in mid-80's [79, 93], the comet assay has been gaining popularity as a standard technique for DNA damage evaluation and genotoxicity testing. It has advantages in terms of sensitivity and the ability to show multiple DNA lesions simultaneously [114], and has been widely used in a variety of applications such as a screening test for breast cancer [85] and a prediction for the risk of bladder cancer [91].

In essence, the comet assay works as follows [24, 114]: Cells treated with a DNA damaging agent (*e.g.*, irradiation) are lysed and loaded onto an agarose gel. An electric field is applied to pull the negatively charged DNA from the nucleus. The DNA is stained with a fluorescent dye, and damaged DNA fragments migrate farther than normal DNA, and relaxed loops and fragments form the tail of a comet, whereas the tightly packed chromatin shows up as its head (Figure 5.2). With increasing dose of the DNA damaging agent, the head of a comet becomes dimmer and its tail grows longer and brighter.

Traditional comet assays suffer from issues such as low throughput, limited reproducibility, and time-consuming and error-prone analysis steps. To alleviate these, new comet assay platforms have been proposed [52, 68, 117, 114]. Their innovations span over various aspects, but the basic analysis principle remains unchanged. In particular, due to overlapping comets and debris, most of the existing analysis programs require manual identification of comets from the fluorescence image. For high-throughput experiments producing a large number of comets, this step bottlenecks the whole analysis pipeline, and there is a clear need for automation.

There have been pioneering attempts to automate high-throughput comet analysis with limited success. A method exists that can perform automated imaging and analysis [114], but it is limited to microwell-array based comet assays that have highly regular structures with predetermined comet locations [114]. A commercial program called CometScore (TriTek Corp., Sumerduck, VA) can handle comet images in arbitrary configurations but is only semi-automated in that users need to specify the boundary of a comet for its automated characterization.

In this paper, we propose DeepComet, a computational tool to facilitate

the analysis of high-throughput comet-assay data. Given a noisy image containing a number of comets placed arbitrarily, DeepCometcan recognize and characterize normal and damaged comets in a fully automated fashion, handling debris and overlaps between comets. Identifying individual comets in the input image is related to the problem of image segmentation [107]. While existing image segmentation techniques tend to show unsatisfactory performance, DeepCometutilizes a suite of new algorithms tailored for recognizing.

After image segmentation, we classify individual objects with convolutional neural networks (CNN) [57] and characterize them by calculating major parameters such as tail moments. CNNs are multi-layer neural networks composed of convolutional, pooling, and fully-connected layers. CNNs achieve recent breakthroughs and are gaining popularity as an end-to-end image recognition module which does not require traditional vision techniques like histogram of oriented gradients (HoG) [20]. To foster widespread use, we provide an online implementation of DeepCometfreely available at http://147.46.126.127:11080/comet/.

The rest of this paper is organized as follows: in Section II we introduce parameters for characterizing comets and review related image processing techniques. In Section III, we present the proposed DeepCometmethodology, explaining the details of each step. Section IV presents our experimental results and discussion. Section V concludes the paper.

5.2 Backgrounds

5.2.1 Understanding comets

Fig. 5.2 shows an example comet image and the parameters used to characterize a comet. Each comet image shows the shape of DNA damage in a single cell

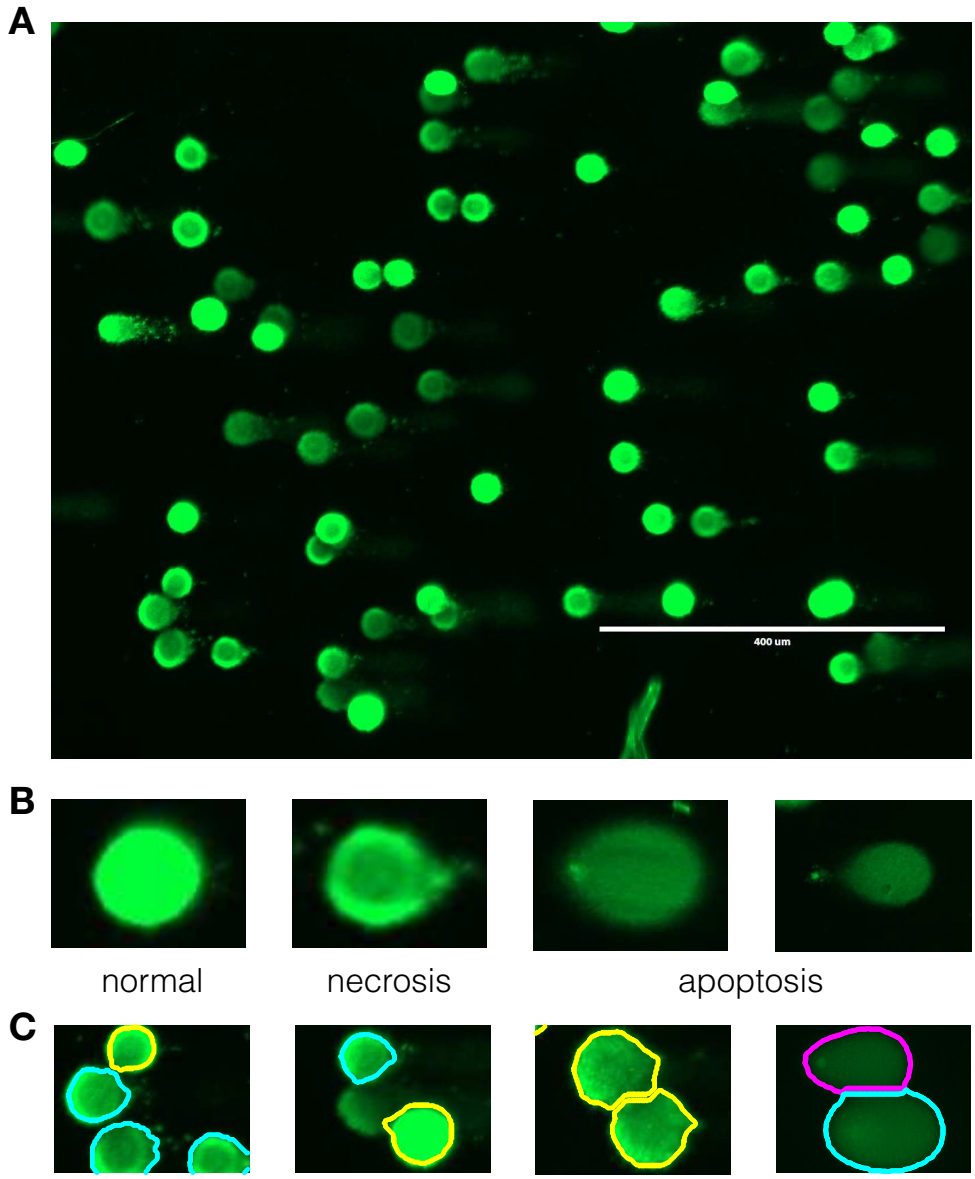

Figure 5.1: Comet assay: (a) DeepCometproduces an output image with multiple comets. (b) Comets are classified into different types according to their shape [42] (yellow: normal, cyan: necrosis, and purple: apoptosis). (c) DeepCometcan automatically correct overlap comets.

and consists of two major parts, namely the head and tail. The intensity and arrangement of pixels conveys information. As damage to the DNA increases, the head becomes dimmer whereas the tail grows longer and brighter [78].

By considering intensity as mass, we can convert the definitions developed in classical physics (such as the center of mass, and the moment of inertia) into the parameters characterizing comets. The bottom plot in Fig. 5.2 introduces the key parameters of a comet image. The x axis corresponds to the horizontal location of pixels, and the y axis indicates the intensity integrated over the vertical direction of the image.

We model the outline of a comet head by a circle. The *center position of head* (CPH) indicates the location of the head center in the x axis and is defined as the peak position in the intensity curve. The user-specified parameter called the *head threshold* (HT) specifies the fraction of the maximum intensity and is used to define the head size. That is, we define the radius r of the head as the distance between CPH and the location whose integrated intensity corresponds to HT [36].

The tail stretches from the right end of the head to the location where the intensity diminishes, and the distance between these two points defines the tail length. We can compute the *center of mass of tail* (CMT) on the x axis, and the tail distance is defined as the difference between CMT and CPH.

It is customary to assume that the total amount of DNA in a comet is proportional to the sum of intensity values of the pixels representing the

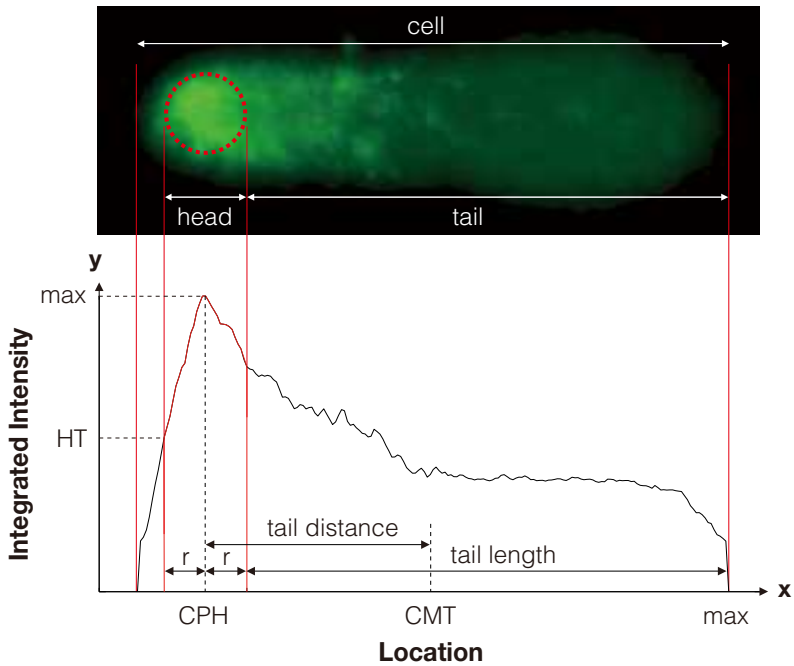

Figure 5.2: Definition of parameters characterizing a comet. Abbreviations: CPH, center position of head; CMT, center of mass of tail; HT, head threshold; r, head radius.

comet [36]. That is,

$$\mathrm{DNA} = \sum_{x \in \mathrm{comet}} I(x) \tag{5.1}$$

$$\mathrm{TDNA} = \frac{1}{\mathrm{DNA}} \sum_{x \in \mathrm{tail}} I(x) \tag{5.2}$$

where DNA and TDNA represent the amounts of the DNA in the cell (represented by the entire comet) and in the tail, respectively, and $I(x)$ the intensity of the pixel at x.

5.2.2 Assessing DNA damage from tail shape

There are several different measures that quantify the degree of DNA damage implied by the tail image. The simplest way is to consider the amount of DNA

in the tail and its length together, which defines the (tail) *extent moment* [18]:

$$\text{extent moment} = \text{TDNA} \times \text{tail length} \tag{5.3}$$

Note that the higher the extent moment, the more DNA damage. One limitation is that it is difficult to differentiate comets with the identical TDNA and tail length but of different shape.

To resolve this issue, Olive proposed the (tail) *Olive moment* as follows [78]:

$$\text{Olive moment} = \text{TDNA} \times \text{tail distance} \tag{5.4}$$

which includes the center of mass of the tail into the calculation by using the tail distance instead of the tail length.

Furthermore, we can consider the distribution of tail pixels by using the *moment of inertia* of the tail [36] defined as

$$\text{moment of intertia} = \frac{1}{\text{DNA}} \sum_{x \in \text{tail}} I(x) \times (\text{CPH} - x)^2 \tag{5.5}$$

where the last term represents the squared distance between the center of the head and each pixel in the tail.

5.2.3 Related image processing techniques

We briefly review image segmentation techniques used in the proposed method. Fig. 5.3 shows a taxonomy of existing image segmentation techniques [107]. The problem of image segmentation concerns how to recognize and extract objects embedded in a background image. Broadly, there are two main types of approaches to the image segmentation problem, namely spatially blind and

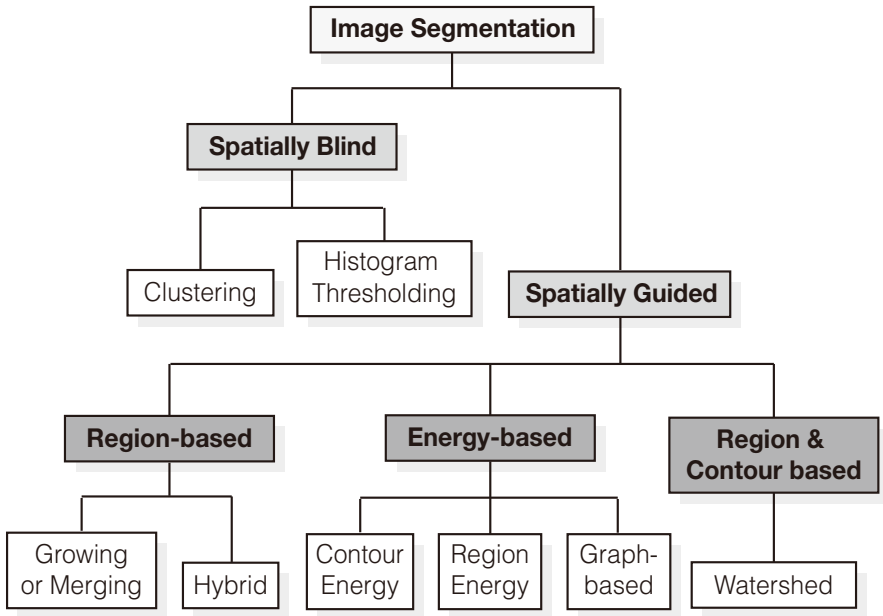

Figure 5.3: Taxonomy of image segmentation methods [107].

spatially guided approaches, depending on the need for providing additional information (such as the gradient of regions and edges in the image). In the present work, which aims at fully automated identification of comets, spatially blind methods are better suited. Among those, clustering-based techniques have the advantages of simplicity and ease of implementations but suffer from the limitations that it is often difficult to determine the right number of clusters due to the dependence of point clouds on the input image. In this work, the number of clusters is fixed to two (either comets or the background), and as will be compared with the proposed segmentation, the 2-means clustering method tends to result in under-segmentation.

In order to detect and correct overlap comets, we further employ the watershed algorithm [7] and the Fourier shape descriptor [116]. The watershed method is an elegant segmentation tool based on morphological shapes [50]. A gray level image is considered as a topographic relief, and the intensity of a

pixel corresponds to the elevation at the point. The contour of a object (called the watershed) can be determined as the limits of the catchment basins of the drops when drops of water flow on a topographic relief. There are several methods to achieve the watershed principle; In this paper, we utilize the well-known Meyer's algorithm [7] followed by distance transform [51] of a binary image. Overlapping comet areas can be divided into several chunks through the watershed, and we test whether each chunk is a valid comet or not by using the Fourier descriptor.

For classification of individual comets, we exploit convolutional neural networks (CNN), one of the widely used classifiers for images. While traditional approaches extract image features and train classifiers separately, CNNs learn both image filters and classifiers simultaneously. Especially, the convolutional operation, the key idea of CNN, enables us to obtain discriminative filters taking advantage of 2D structure in images. In our experiments, CNN-based end-to-end image recognition outperformed a combination of HoG features and Support Vector Machine (SVM).

5.3 Methods

Fig. 5.4 shows the overall flow of the proposed methodology. The input is an image in 8-bit RGB format containing multiple comets obtained from a high-throughput comet-assay experiment with no limitations on comet locations or quantities. The proposed methodology does not assume a specific configuration of comet locations, which is an important advantage over existing softwares. DeepCometconsists of 4 major steps: preprocessing, binarization, filtering, and characterization. Through all the steps, DeepCometproduces a set of identified

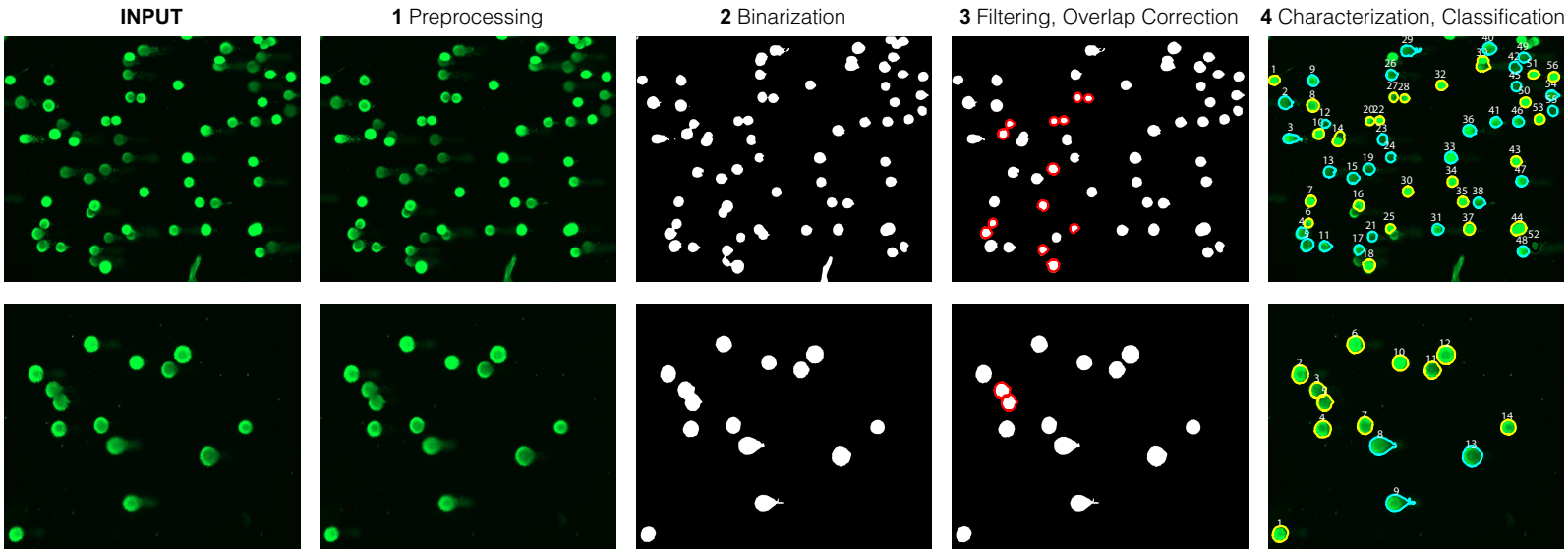

Figure 5.4: Overview of proposed DeepComet methodology.

comets with their properties. The rest of this section explains each step of the process in more detail.

5.3.1 Preprocessing

This step carries out a smoothing procedure such as median filtering or moving-average filtering. By using a moving window, every pixel in the window is replaced with the median or average value within the window. This blurring operation can reduce noise, thus facilitating downstream processes. In particular, we observed that the blurring operation could improve the accuracy of the thresholding for the binarization and could decrease erroneous dissection of the head and the tail.

5.3.2 Binarization

The purpose of this step is to distinguish and separate objects from the background. The pixel intensity of comet assay images corresponds to a density of fragments of the cells. Thus, we exploit simple thresholding method. One of the well-known thresholding is Otsu's method. The algorithm carries out minimization of the within-class variance σ_W^2 , or alternatively maximization of the between-class variance σ_B^2 defined by the following equations [80]:

$$\begin{aligned} \sigma_W^2 &= \omega_0\sigma_0^2 + \omega_1\sigma_1^2, \\ \sigma_B^2 &= \omega_0\omega_1(\mu_1 - \mu_2)^2, \end{aligned}$$

where notations ω_i, μ_i, and σ_i denote the probability of occurrence, the mean intensity, and the variance of intensity values for class i, respectively. Otsu's method has been working well for high contrast images [60], however did not

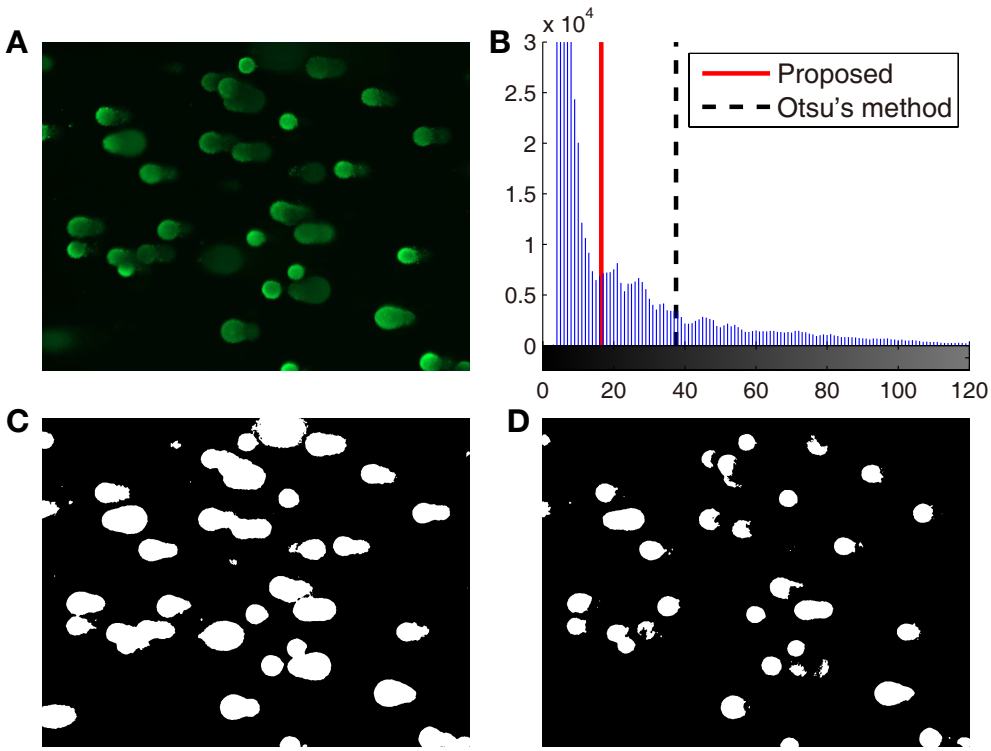

Figure 5.5: Binarization example: (a) original image. (b) intensity histogram of grayscale image. (c) proposed binarization. (d) Otsu's binarization.

perform properly for comet assay images due to the variance of intensites in comet pixels.

As shown in Fig. 5.5(a), comet pixels are presented with a wide intensity range. In this case, Otsu's method hardly find out faint areas which are mostly abnormal cells that should be catched (see Fig. 5.5(d)). For this under segmentation problem, we draw a gray level histogram and seek the first valley point as indicated in Fig. 5.5(b). Gray level histograms of comet assay images always have the first peak at the background intensity. While Otsu's method results in a high threshold because it minimizes the variance of intensities for comets, the first valley is put in somewhere between the background and Otsu's threshold. Thus, relying on the first valley of a histogram gives the effect of using an adaptive threshold to distinguish comet and background pixels as illustrated in Fig. 5.5(b).

Because the first valley detection would give over segmentation results (see Fig. 5.5(c)), the filtering described in the next subsection will focus removal of false positives so that individual comets have elaborate contours. After this preliminary segmentation, DeepCometperforms adjacent pixel grouping based on 8-pixel connectivity and labels the identified comet candidates. This step completes the first round of comet identification.

5.3.3 Filtering and overlap correction

Out of the identified comets in the previous step, DeepCometdiscards "incomplete" comets for further analysis. Recall that the aim of the previous step is to report all the possible comet areas although false comet areas are included in the candidates. In this step, DeepCometfirst removes the comets lying across the boundaries of the input image because most of them have invalid shapes.

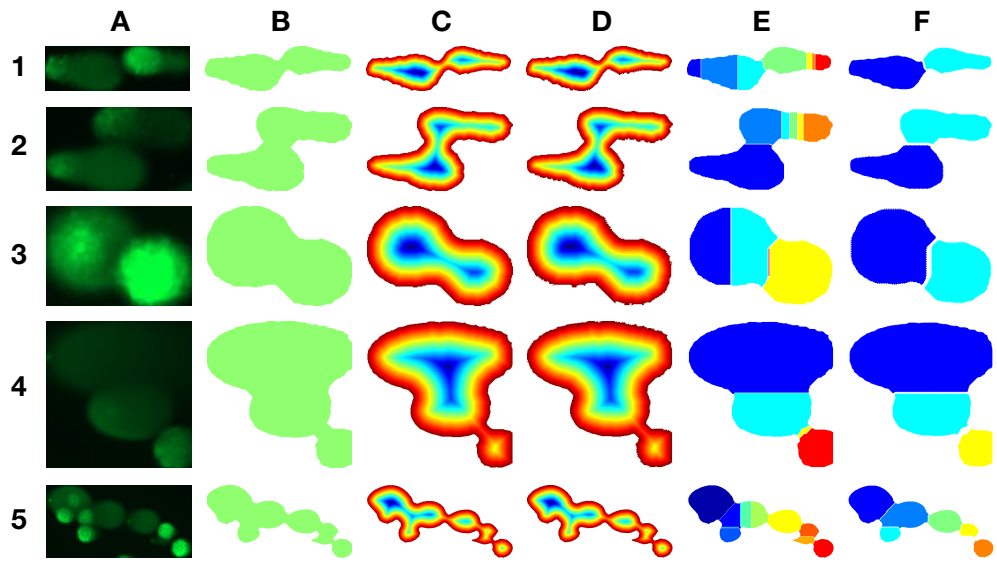

Figure 5.6: Overlap correction process (best viewed in color): (a) original image. (b) initial mask. (c) distance transform of (b). (d) wavelet transform of (c). (e) watershed transform of (d). (f) merging and filtering.

DeepCometthen identifies overly small groups of pixels, namely those whose number of pixels is less than a threshold (*e.g.*, 0.1% of the total pixels in the input image). Such tiny groups are not removed immediately but are merged into the closest comets. According to our experiments, such tiny groups play a role in shaping certain types of comets (*e.g.*, the apoptosis type in Fig. 5.1(b)).

After the filtering step, we detect overlapping comets and corrects them. Fig. 5.6 shows the proposed overlap correction process. About five overlap examples as seen in Fig. 5.6(a), we obtained those initial masks as depicted in Fig. 5.6(b) in the previous step. To identify the number of morphological shapes in the masks, we perform the watershed transform followed by the distance transform. The watershed operator can detect multiple overlapping shapes in the ideal case of smooth contour. However in reality, the overlapping comets have noisy borders and individual comets among those have also irregular shapes.

In order to address the robustness of watershed, we apply the wavelet transform after the distance transform. The distance transform works as a generator of a topographic relief (see Fig. 5.6(c)). Each pixel has an altitude which is calculated as the distance to the nearest boundary pixel in a binary image. This topographic relief may have many shallow holes which cause over-segmentation when performing the watershed transform. Thus, we utilize the wavelet transform as a smoothing filter of the topographic map as shown in Fig. 5.6(d).

We then obtain candidates of individual comets in the original binary image by using the watershed transform (see Fig. 5.6(e)). However, there are still over-partitioned chunks. This problem can be solved easily. We only need to merge horizontally divided areas into one segment, because the horizontal division arises by irregular contour of one shape. Fig. 5.6(f) shows the results through a series of processes so far, except for the final filtering step.

After the horizontal merging, we have to check validity of each chunk. We assumed that all the cells have elliptical shapes, and exploited the Fourier shape descriptor [116] to decide roundness of each cell. Fig. 5.7 shows a characteristics of the Fourier descriptor with 9 different shapes. We can see an object on frequency domain with the Fourier transform of 2D coordinates of contour points. Then, low-order frequencies are more dominant as cell shapes tend to be elliptical. Based on this observation, we established two criterions to decide a validity of each chunk. First, we check if the absolute sum of amplitudes of the two lowest frequencies is greater than 70% of the absolute sum of all the frequencies. Second, we discard a chunk if its area over the area of the initial mask is lower than 3%. Then, we can obtain the partitioned segments as shown in Fig. 5.6(f).

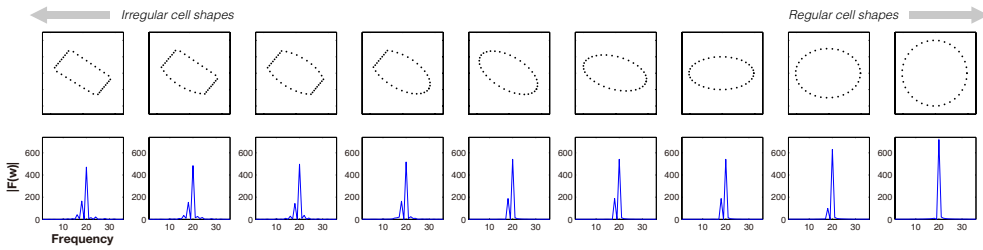

Figure 5.7: Fourier descriptors on 9 different shapes. Cells generally have elliptical shapes in which low-order frequencies are dominant.

5.3.4 Characterization and classification

The final step is to characterize each comet to find its parameters such as the sizes of comets and their heads and tails and the tail moments. In particular, the tail moment plays a key role to assess the degree of DNA damage of a cell. DeepCometreports three types of tail moments: the extent and the Olive moments, and the moment of inertia, as defined in Section 5.2. As an example of the comet characterization, Fig. 5.8 shows the distribution of some 300 comets in terms of the extent moment and the width/height ratio.

After the characterization, we extract *histogram of oriented gradients* (HOG) [20] features from each cell image. HOG is a feature descriptor used in computer vision and image processing for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image. We compare 4 classifiers with HoG features and the experimental results are described in the next section.

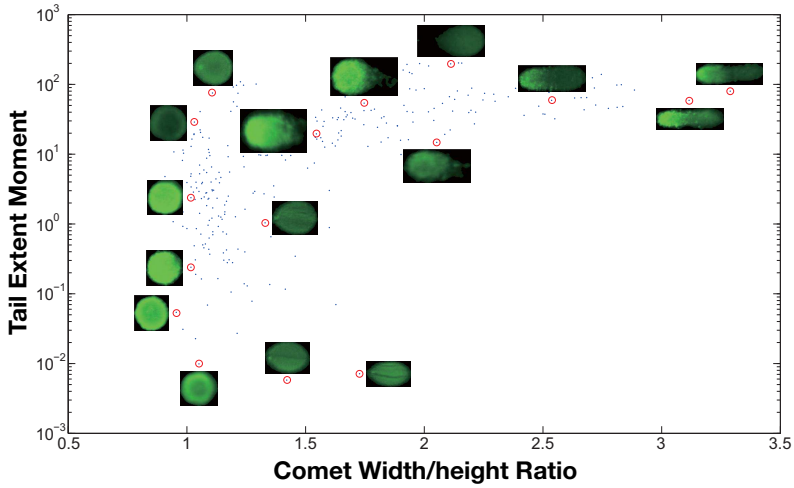

Figure 5.8: Characterization of comets. The x-axis and y-axis represent the width/height ratio and the tail extent moment, respectively. This plot characterizes some 300 comets sampled from the 35 test images (Table 5.1), each of which appears as a blue dot and some are accompanied with comet images for visual inspection.

5.4 Results and discussion

5.4.1 Test data preparation

To evaluate the performance of DeepComet, we tested it with 35 golden data sets verified by domain experts. Each data set is based on the image from a micro comet-assay system (PICASSo, currently under development, NanoEnTek Inc., Korea). The system consists of a gel-electrophoresis micro-chamber and parallel multi-microchannels, which enables loading of a low melting-point agarose (LMA) gel mixed with single cells. In these experiments, Jurkat cells were exposed to toxic material (20 mM hydrogen peroxide) for 10 min and loaded into multi-microchannels. After electrophoresis and nucleic acid staining with SYBR green, fluorescent images were captured by a microscope (EVOS, AMG Inc., USA). Three domain experts visually identified comets from each of the 35 images, reporting 8–56 comets per image (20.03 on aver-

age and 702 in total). The throughput for processing these test images was over 1000 comets per minute. Further details of each of the sample images are listed in Table 5.1.

5.4.2 Binarization

We evaluated binarization performance over the 35 test images with a region-based measure. Each segmentation method labels each pixel of the image with a binary value for whether a comet pixel or not. We regarded a binarization of a comet assay image as a binary vector, and calculated true positives (TPs), false positives (FPs), and true negatives (TNs). TP (TN) refers to a comet (background) pixel that is correctly binarized as a comet (background). FP is a background pixel that is incorrectly binarized as a comet. For each image, we calculated precisions and recalls[1].

We compared the proposed binarization with three alternatives: Otsu's method [80], K-means [34], and GraphCut [9]. Fig. 5.9 shows the precision, recall, and runtime of four methods on all the test images. DeepCometresulted in 15.8% lower precision values than the GraphCut on average (0.646 versus 0.798). However, DeepCometoutperformed the three alternatives in terms of recall, yielding up to 21.3% higher average recall value than the GraphCut.

In the binarization, we aimed to perform preliminary segmentation for the following enhancement procedure. As minimizing FNs, incorrectly segmented actual positives, we can preserve true comet areas in the first round and concentrate on minimizing of FPs in the next filtering step. The proposed thresholding not only identified actual comet pixels successfully (0.979 recall on average) but also did perform in 0.1 seconds. Since the boundary of comets

[1] $\text{precision} = \text{TP}/(\text{TP} + \text{FP}),\ \text{recall} = \text{TP}/(\text{TP} + \text{FN}).$

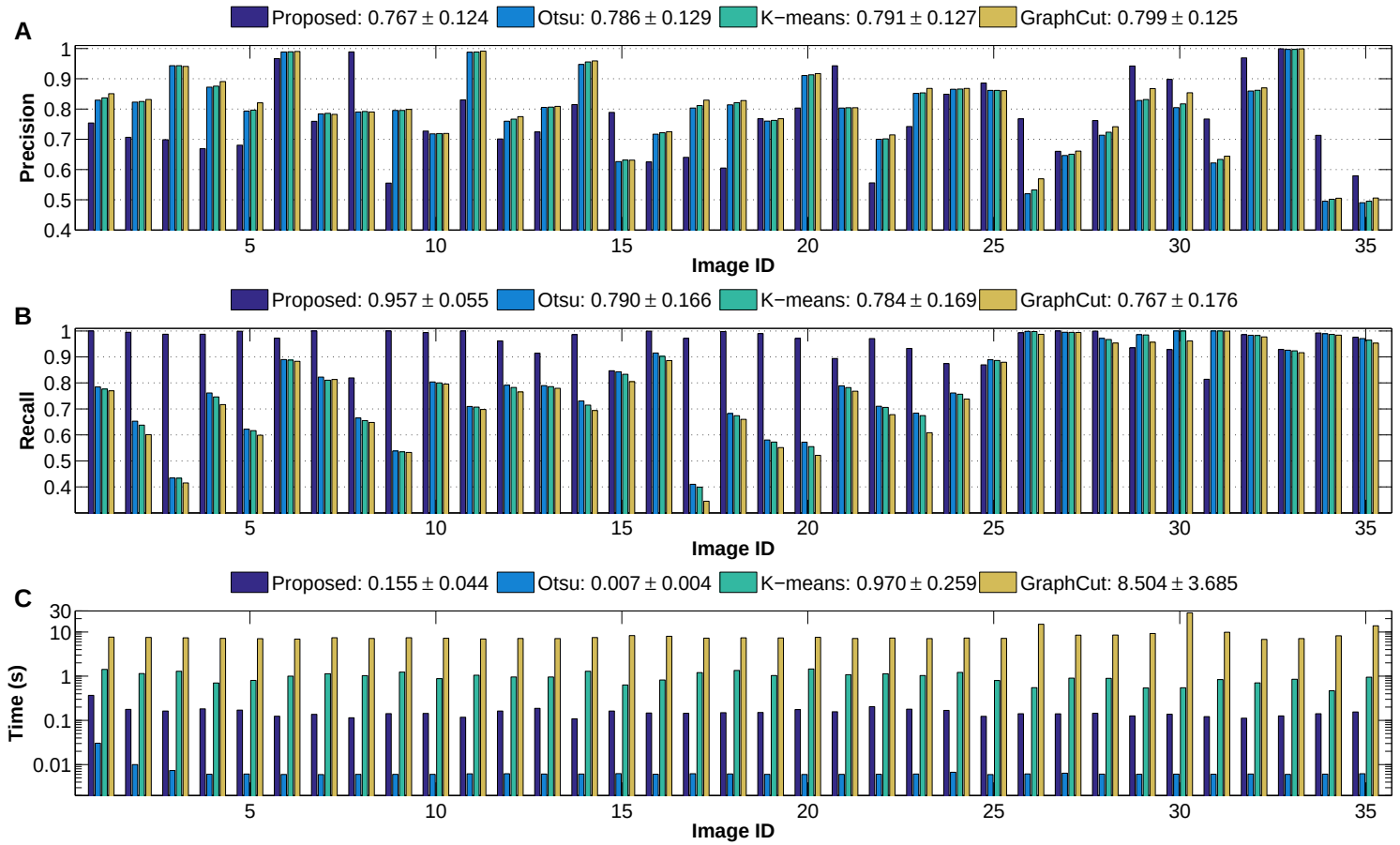

Figure 5.9: Comparison of binarization performances in terms of (a) precision, (b) recall, and (c) runtime.

are smeared and blurry due to DNA fragments, the three alternatives that focus on detecting objects with crisp boundaries overly missed fragment pixels around the main body. This is understandable given that the these methods is designed for general images in which people normally want clean-cut image segmentation.

5.4.3 Robust identification of comets

By comparing the comets identified by DeepCometto those identified in the reference identification procedure, we count the numbers of TPs, FPs, and FNs for each test image. As counted the correctly detected cells, we calculated with *centroid-based measure* different from the *region-based measure* in the previous subsection. A reported cell would be counted as a TP if its centroid is located within a range of 15 pixels (= 12μm) from the centroid of a ground truth cell. Based on these numbers, we calculated precisions, recalls, and F1-scores for each image as listed in Table 5.1. The F1-score is defined by a harmonic mean of a precision and a recall.

Through the filtering and overlap correction (Step 3), DeepCometraised the average precision and recall values from 0.74 and 0.73 to 0.93 and 0.92, respectively. For GraphCut, the average precision and recall were 0.75 and 0.66 despite of applying the correction procedure. The four methods described in the previous subsection are ranked according to the centroid-based F1-score as the adaptive thresholding (0.92), Otsu's method (0.77), K-means (0.76), and GraphCut (0.71). Even if we utilized the three alternatives without the filtering, we could obtained up to 0.6 average F1-scores. Given comet assay images in which object boundaries are cloudy, we believe that the proposed thresholding and correction schemes should be used together to recognize all

Table 5.1: Details of 35 test images

Details of 35 Test Images

Image ID	# of comets	TP	FP	FN	Precision	Recall	F1-score
1	19	19	1	0	0.95	1.00	0.97
2	27	27	3	0	0.90	1.00	0.95
3	22	19	1	3	0.95	0.86	0.90
4	18	14	3	4	0.82	0.78	0.80
5	24	23	0	1	1.00	0.96	0.98
6	14	14	0	0	1.00	1.00	1.00
7	18	16	2	2	0.89	0.89	0.89
8	10	10	0	0	1.00	1.00	1.00
9	11	10	2	1	0.83	0.91	0.87
10	15	15	1	0	0.94	1.00	0.97
11	12	12	1	0	0.92	1.00	0.96
12	40	36	4	4	0.90	0.90	0.90
13	45	38	5	7	0.88	0.84	0.86
14	8	8	0	0	1.00	1.00	1.00
15	14	11	2	3	0.85	0.79	0.81
16	13	12	1	1	0.92	0.92	0.92
17	17	16	1	1	0.94	0.94	0.94
18	21	17	0	4	1.00	0.81	0.89
19	24	20	2	4	0.91	0.83	0.87
20	21	16	1	5	0.94	0.76	0.84
21	42	41	2	1	0.95	0.98	0.96
22	46	37	5	9	0.88	0.80	0.84
23	44	39	6	5	0.87	0.89	0.88
24	56	54	1	2	0.98	0.96	0.97
25	12	11	0	1	1.00	0.92	0.96
26	15	15	1	0	0.94	1.00	0.97
27	9	9	2	0	0.82	1.00	0.90
28	10	10	0	0	1.00	1.00	1.00
29	13	13	0	0	1.00	1.00	1.00
30	11	10	0	1	1.00	0.91	0.95
31	8	7	0	1	1.00	0.88	0.93
32	9	9	0	0	1.00	1.00	1.00
33	13	13	0	0	1.00	1.00	1.00
34	8	6	1	2	0.86	0.75	0.80
35	13	12	3	1	0.80	0.92	0.86
Total	702						
Average	20.06				0.93	0.92	0.92

the comets including blurred and noisy objects.

5.4.4 Classification

To evaluate the performance of DeepComet, we measured the average accuracy of 10-fold cross validation as shown in Table 5.2. Original data consists of over 700 comets having three classes (normal, necrosis, and apoptosis). As CNNs require more weights values than the alternatives, we enlarge the training data set 10 times via data augmentation (vertical flipping and rotation). While grayscale images are fed into CNN, HoG features are used for the five alternatives. We collected two HoG features with the overlapping 8x8 grid from all the resized images (50x50). The boxratio, defined as the width divided by the height of a cell image, is used togetherfFor discrimination of horizontally long types (*e.g.*, necrosis; see Fig. 5.11(d)).

The classifiers compared are *Support Vector Machine* (SVM) [19] with two different kernels, *Neural Networks* (NN) [88], *AdaBoost* [25], and *Classification And Regression Trees* (CART) [10]. Details of the individual classifiers are described as follows. For SVM, we trained three binary classifiers (normal vs. necrosis, normal vs. apoptosis, and necrosis vs. apoptosis) and aggregate the three classifiers as a decision tree, which is a traditional approach for multiclass SVM. We tried two kernels: radial basis function (rbf) and linear. For NN, we constructed only input and output layers composed of 3 nodes with softmax regression. For AdaBoost, we chose a decision tree as a template classifier and exploited 200 trees. For CART, the maximum pruning level was set to 10. Lastly, for CNN, we trained 10 networks and ensemble the results of 10 different models.

The linear SVM showed the best predictive accuracy 89.291% using original data. Under small number of training images (*e.g.*, 700), CNN did not exhibit satisfactory performance compared to linear and rbf kernel SVMs. However,

Table 5.2: Classification performances. Hog features are fed into classifiers except that grayscale images are fed into CNNs.

	original data	augmented data (x10)
CNN	86.532 ± 4.636	$\mathbf{90.272 \pm 2.754}$
SVM (rbf)	89.122 ± 3.574	89.775 ± 2.714
SVM (linear)	$\mathbf{89.291 \pm 3.792}$	86.380 ± 3.994
NN	82.020 ± 4.695	74.894 ± 4.374
AdaBoost	81.531 ± 4.370	74.561 ± 6.293
CART	79.730 ± 4.827	76.655 ± 6.136

CNN outperformed the SVMs, achieving 90.272% accuracy when the size of the training data was increased. The rbf SVM produced 0.653% improved but the linear SVM resulted in the degraded performance via data augmentation, because the enlarged HoG features are not linearly separable.

After the classification, DeepCometcan identify around 90% of non-overlapped cells in one comet assay image. In order to evaluate DNA damages correctly, we should extract *heterogeneity of response* [35, 69, 95] (*e.g.*, a distribution of % DNA in tail). When the true distribution of % DNA in tail is normal, discarded 10% cells would result in only 1.7% decreased confidence level with the same confidence interval (*e.g.*, 95% confidence interval using 50 comets = 93.7% confidence interval using 41 comets). Thus, DeepCometis able to provide a satisfiable performance given sufficient comets in one image (*e.g.*, 25 comets [35]).

5.4.5 More accurate characterization by DeepComet

After a comet is recognized, we should characterize it by measuring key parameters such as tail moments. We compared the tail moments calculated by DeepCometand an existing program called CometScore (TriTek Corp., Sumerduck, VA). Fig. 5.10 shows the correlation of the Olive moment and the extent

moment between the two methods. For both types of moments, the correlation was positive. The correlation of the Olive moment (0.8446) was higher than that of the extent moment (0.6026).

The discrepancy is mainly due to the difference in defining the head of a comet. CometScore assumes that the diameter of the head of a comet is identical to the height of the comet. This assumption is reasonable for certain cases (*e.g.*, normal and necrosis; see Fig. 5.11), but fails to model the comet shapes in other cases (*e.g.*, apoptosis; see Fig. 5.11).

Table 5.3: Details of Comet Images Shown in Fig. 5.11(c) and Fig. 5.11(d)

method A: DeepComet B: CometScore		comet length (px)	comet height (px)	head diameter (px)	tail length (px)	tail dist.	TDNA (%)	extent moment (%)	Olive moment (%)
Fig. 5.11(c)	A	84	46	5	78	48	99.53	77.64	47.78
	B	84	49	49	35	46	74.84	26.19	34.94
Fig. 5.11(d)	A	82	67	13	58	28	95.6	55.46	26.77
	B	100	74	74	26	33	65	16.92	21.07

Consequently, CometScore tends to overestimate the diameter of a head in case of apoptosis comets, producing underestimated tail moments. This fact is reflected in the correlation plots in Fig. 5.10, where the Olive or extent moment values calculated by DeepCometend to be higher. The difference was more noticeable for the extent moment than for the Olive moment. The reason comes from the definitions of the two moments. As defined in Eq. (5.3), the extent moment is the product of TDNA (the fraction of total DNA contained in the tail) and the tail length (the distance between the head boundary to the end of the tail). If we overestimate the head size, then the tail size becomes smaller than the actual value. This lowers both the TDNA and tail length values, producing an underestimated extent moment value.

In comparison, the Olive moment is defined as the product of TDNA and

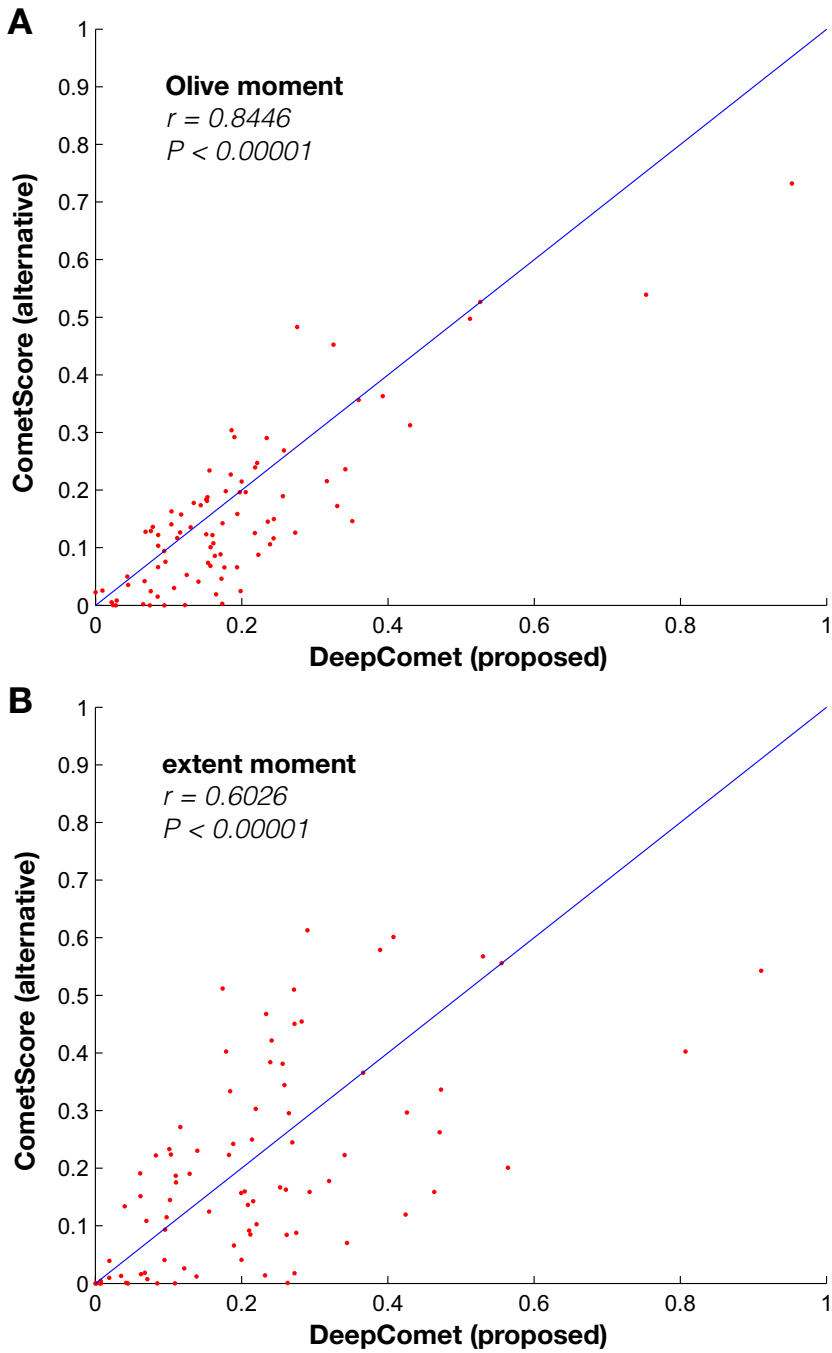

Figure 5.10: Correlation of normalized tail moments between two tools: DeepCometand CometScore (TriTek Corp., Sumerduck, VA). 86 comets were randomly sampled from the 40 test images used. (a) Olive moment. (b) extent moment.

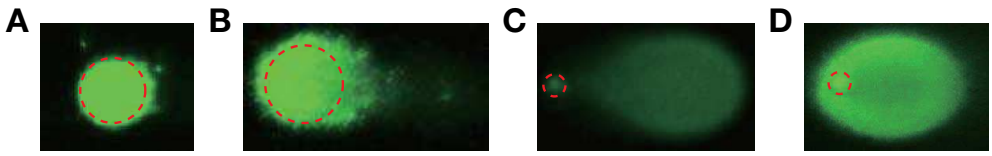

Figure 5.11: Comets and their heads. Green images represent comets, on which red circles indicate the head locations determined manually by domain experts. The states represented by the four images are as follows: (a) normal (b) necrosis (c) and (d) apoptosis. For (a) and (b), the head diameters are close to the comet heights. In contrast, the head diameter is smaller than the comet heights. CometScore (TriTek Corp., Sumerduck, VA) does not consider this fact and tends to overestimate head diameters, resulting in underestimated tail moments. More details of (c) and (d) are listed in Table 5.3.

the tail distance (the distance from the head center to the center of mass of the tail). Thus, for the Olive moment, overestimating the head size lowers the TDNA value, but often increases the tail distance due to the shift of the CMT towards the end of the tail. These two affect the Olive moment calculation in the opposite direction. Consequently, the underestimation of the Olive moment by CometScore tends to be less significant than that of the extent moment, and the Olive moment values estimated by CometScore show higher correlation with DeepCometthan for the extent moment ($r = 0.8446$ versus $r = 0.6026$ in Fig. 5.10a). This observation also confirms the robustness of the Olive moment, compared with the extent moment.

5.5 Summary

We have presented DeepComet, an automated tool for high-throughput comet-assay analysis. The key features of DeepCometinclude the following. First, DeepCometautomatically recognizes individual comets from the input image without making any assumption on the number of comets or their locations. This is critical for reducing the time demands of analyzing high-throughput

assays with many comets. Second, DeepCometcan detect overlapping comets and isolate them. Without this feature, experimentalists have no other choice but to discard overlapping comets even though the front comets are completely eligible for analysis. Given that overlaps occur frequently in typical high-throughput comet assays, this functionality should be useful for maintaining sufficient comet counts for analysis by salvaging parts of overlapping comets. Third, DeepCometcan characterize each of the recognized comets with convolutional neural networks and report key parameters such as tail moments without making overly simplified assumptions on comet shapes as certain existing tools do. Given the effectiveness of DeepComet, it is our hope that DeepCometcan more greatly facilitate high-throughput comet-assay analysis by accelerating its most rate-limiting steps.

Chapter 6

Conclusion

As deep neural networks learn a large number of parameters, there have been many attempts to obtain reasonable solutions over wide search spaces. In this dissertation, three issues facing deep learning are discussed along with possible solutions using regularization techniques. This final chapter summarizes our contributions and presents possible future research directions.

6.1 Dissertation summary

Chapter 1 and 2 introduce deep neural networks and the three main issues, followed by background materials. Chapter 3, 4, and 5 describe individual approaches for these three issues.

- Chapter 3 described a methodology unifying deep learning and manifold learning. Motivation for the unification came from the character of adversarial examples. We utilize those examples in interpolations between the nearest training samples on manifold embedding space. We call this framework manifold regularized networks (MRnet), because we

considered an additional loss term that measures how dissimilar original and adversarial samples are on low dimensional manifold space. Traditional neural networks, even state-of-the art deep networks, have intrinsic blind spots due to their huge number of parameters and the linear function components using them. We tested MRnet and confirmed its improved generalization performance, which was underpinned by the proposed manifold loss term on deep architectures. By exploiting the characteristics of blind spots, the proposed MRnet can be extended to the discovery of true manifold representations for various learning tasks.

- Chapter 4 showed the first application of deep belief networks to identifying junction splicing signals. Splicing refers to the elimination of non-coding regions in transcribed pre-messenger ribonucleic acid (RNA). Discovering splice sites is an important machine learning task that helps us not only to identify the basic units of genetic heredity but also to understand how different proteins are produced. Existing methods for splicing prediction have produced promising results, but often show limited robustness and accuracy. This chapter presents an improvement in contrastive divergence when training an RBM for DNA sequences, and more generally, for binary representations of categorical information. In our experiments, our approach achieved F1-scores nearly 20% higher than the alternatives and was particularly effective for training in class imbalanced problems. We presented our work in the context of genomics, but it is applicable to learning with other types of data that contains categorical features.

- Chapter 5 covered a deep learning based automation tool for high-

throughput comet-assay analysis called *DeepComet*. DeepComet automatically recognizes individual comets from the input images without making any assumption about the number of comets or their locations. DeepComet also can detect overlapping comets and isolate them. After segmentation and overlap detection, DeepComet can characterize each of the recognized comets with convolutional neural networks (CNN) and report key parameters, such as tail moments, without making overly simplified assumptions on comet shapes, as certain existing tools do. To recognize comets with CNN, we augmented individual comet images due to the insufficient amount of data, which is one of the major issues facing biomedical image processing. Given the effectiveness of DeepComet, it is our hope that DeepComet can facilitate high-throughput comet-assay analysis by accelerating its most rate-limiting steps.

In summary, this dissertation proposed a set of deep learning regularization schemes that can learn meaningful representations underlying large-scale genomic datasets and image datasets. The effectiveness of these methods was confirmed with a number of experimental studies.

6.2 Future work

There are several future research possibilities stemming from the proposed methodologies. First, we are able to extend MRnet to extract scaling and translation invariant features by replacing synthetic nearest training samples. In essence, MRnet minimizes the 2-norm difference between original and adversarial examples in manifold space. Similarly, we can expect that MRnet would capture translation invariant features by minimizing the difference be-

tween the original and the translations. Second, it can also be interesting to alternate the objective function of MRnet in order to generalize the MRnet procedure. Currently, MRnet requires two forward-backward operations. The second forward-backward phase may be unnecessary because adversarial perturbations are derived during the first backward operation. By slightly modifying the proposed manifold loss, the two forward-backward steps can be combined into one. Lastly, the proposed schemes (manifold loss and boosting) can be applied into the framework of recurrent neural networks. In this dissertation, we mainly focused on fully connected models and convolutional neural networks. As the proposed methods are for general neural networks, we can anticipate that the methods can be extended to recurrent neural networks.

Bibliography

[1] Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5(1):13–51, 1995.

[2] Kin Fai Au, Hui Jiang, Lan Lin, Yi Xing, and Wing Hung Wong. Detection of splice junctions from paired-end RNA-seq data by SpliceMap. *Nucleic Acids Research*, 38(14):4570–4578, 2010.

[3] Pierre Baldi and Søren Brunak. Chapter 6. neural networks: applications. In *Bioinformatics: The Machine Learning Approach*. MIT press, 2001.

[4] Abdul KMA Baten, Bill CH Chang, Saman K Halgamuge, and Jason Li. Splice site identification using probabilistic parameters and SVM classification. *BMC Bioinformatics*, 7(Suppl 5):S15, 2006.

[5] Yoshua Bengio. Learning deep architectures for AI. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

[6] Yoshua Bengio, Aaron Courville, and Pierre Vincent. Representation learning: A review and new perspectives. *IEEE TPAMI*, 35(8):1798–1828, 2013.

[7] Serge Beucher and Fernand Meyer. The morphological approach to segmentation: the watershed transformation. *OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED-*, 34:433–433, 1992.

[8] Hervé Bourlard and Yves Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988.

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[10] Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees*. CRC press, 1984.

[11] Søren Brunak, Jacob Engelbrecht, and Steen Knudsen. Prediction of human mRNA donor and acceptor sites from the DNA sequence. *Journal of Molecular Biology*, 220(1):49–65, 1991.

[12] M Burset, IA Seledtsov, and VV Solovyev. Analysis of canonical and non-canonical splice sites in mammalian genomes. *Nucleic Acids Research*, 28(21):4364–4375, 2000.

[13] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: a library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):27, 2011.

[14] Chenyi Chen, Ari Seff, Alain Kornhauser, and Jianxiong Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In

Proceedings of the IEEE International Conference on Computer Vision, pages 2722–2730, 2015.

[15] KyungHyun Cho, Tapani Raiko, and Alexander Ilin. Parallel tempering is efficient for learning restricted Boltzmann machines. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2010.

[16] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Proceedings of Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[17] Yongjun Chu and David R Corey. RNA sequencing: platform selection, experimental design, and data interpretation. *Nucleic Acid Therapeutics*, 22(4):271–274, 2012.

[18] Andrew R. Collins. The comet assay for DNA damage and repair. *Molecular Biotechnology*, 26:249–261, 2004.

[19] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[20] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[21] Sven Degroeve, Yvan Saeys, Bernard De Baets, Pierre Rouzé, and Yves. Van de Peer. SpliceMachine: predicting splice sites from high-

dimensional local context representations. *Bioinformatics*, 21(8):1332–1338, 2005.

[22] Joseph C Dunn. A fuzzy relative of the isodata process and its use in detecting compact well-separated clusters. *Journal of cybernetics*, 3(3):32–57, 1973.

[23] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 11:625–660, 2010.

[24] Daryl W. Fairbairn et al. The comet assay: a comprehensive review. *Mutation Research/Reviews in Genetic Toxicology*, 339(1):37–59, February 1995.

[25] Wei Fan, Salvatore J Stolfo, and Junxin Zhang. The application of adaboost for distributed, scalable and on-line learning. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 362–366. ACM, 1999.

[26] K. Fukushima. Neocognitron: a self organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol Cybern*, 36(4):193–202, 1980.

[27] Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

[28] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech

corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93, 1993.

[29] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[30] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[31] Ian Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron Courville, and Yoshua Bengio. Maxout networks. In *ICML*, pages 1319–1327, 2013.

[32] Gregory R. Grant, Michael H. Farkas, Angel D. Pizarro, Nicholas F. Lahens, Jonathan Schug, Brian P. Brunk, Christian J. Stoeckert, John B. Hogenesch, and Eric A. Pierce. Comparative analysis of RNA-Seq alignment algorithms and the RNA-Seq unified mapper (RUM). *Bioinformatics*, 27(18):2518–2528, 2011.

[33] Ravi Gupta, Ankush Mittal, Kuldip Singh, Prateek Bajpai, and Suraj Prakash. A time series approach for identification of exons and introns. In *Proceedings of International Conference on Information Technology (ICIT)*, pages 91–93, 2007.

[34] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Applied statistics*, 28(1):100–108, 1979.

[35] Janet M Hartley, Victoria J Spanswick, and John A Hartley. *Measurement of DNA damage in individual cells using the single cell gel electrophoresis (comet) assay*, pages 309–320. Springer, 2011.

[36] Björn Hellman, Hamid Vaghef, and Bernt Boström. The concepts of

tail moment and tail inertia in the single cell gel electrophoresis assay. *Mutation Research*, 336(2):123–131, 1995.

[37] Geoffrey Hinton and Terrence Sejnowski. *Learning and relearning in Boltzmann machines*, pages 282–317. MIT Press, 1986.

[38] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[39] Geoffrey E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[40] Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint*, arXiv:1207.0580, 2012.

[41] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[42] Jan H J Hoeijmakers. DNA damage, aging, and cancer. *The New England Journal of Medicine*, 361(15):1475–85, October 2009.

[43] J. Huang, T. Li, K. Chen, and J. Wu. An approach of encoding for prediction of splice sites using SVM. *Biochimie*, 88(7):923–929, 2006.

[44] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[45] AG Ivakhnenko. Polynomial theory of complex systems. *Systems, Man and Cybernetics, IEEE Transactions on*, (4):364–378, 1971.

[46] Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. What is the best multi-stage architecture for object recognition? In *ICCV*, pages 2146–2153, 2009.

[47] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678, 2014.

[48] W. James Kent, Charles W. Sugnet, Terrence S. Furey, Krishna M. Roskin, Tom H. Pringle, Alan M. Zahler, Haussler, and David. The human genome browser at UCSC. *Genome Research*, 12(6):996–1006, 2002.

[49] Hadas Keren, Galit Lev-Maor, and Gil Ast. Alternative splicing and evolution: diversification, exon definition and function. *Nature Reviews Genetics*, 11(5):345–355, 2010.

[50] KyungSu Kim, Jeongjun Song, Forouzan Golshani, and Sethuraman Panchanathan. Automatic classification of cells using morphological shape in peripheral blood images. In *Information Technologies 2000*, pages 290–298. International Society for Optics and Photonics, 2000.

[51] Ron Kimmel, Nahum Kiryati, and Alfred M Bruckstein. Sub-pixel distance maps and weighted distance transforms. *Journal of Mathematical Imaging and Vision*, 6(2-3):223–233, 1996.

[52] Evangelos Kiskinis, Willi Suter, and Andreas Hartmann. High throughput Comet assay using 96-well plates. *Mutagenesis*, 17(1):37–43, 2002.

[53] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Report, 2009.

[54] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[55] Karol Kurach, Marcin Andrychowicz, and Ilya Sutskever. Neural random-access machines. In *ICLR*, 2016.

[56] Charles E Lawrence, Stephen F Altschul, Mark S Boguski, Jun S Liu, Andrew F Neuwald, and John C Wootton. Detecting subtle sequence signals: a Gibbs sampling strategy for multiple alignment. *Science*, 262(5131):208–214, 1993.

[57] Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[58] Honglak Lee, Chaitanya Ekanadham, and Andrew Y. Ng. Sparse deep belief net model for visual area V2. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 873–880, 2008.

[59] Honglak Lee, Peter T. Pham, Yan Largman, and Andrew Y. Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1096–1104, 2009.

[60] Graham Leedham, Saket Varma, Anish Patankar, and Venu Govindaraju. Separating text and background in degraded document images-a comparison of global thresholding techniques for multi-stage threshold-

ing. In *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*, pages 244–249. IEEE, 2002.

[61] Pietro Di Lena, Pierre Baldi, and Ken Nagata. Deep spatio-temporal architectures and learning for protein structure prediction. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 521–529, 2012.

[62] Michael KK Leung, Hui Yuan Xiong, Leo J Lee, and Brendan J Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

[63] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *ICLR*, 2014.

[64] David J Lockhart and Elizabeth A Winzeler. Genomics, gene expression and DNA arrays. *Nature*, 405(6788):827–836, 2000.

[65] Jiwen Lu, Gang Wang, Weihong Deng, Pierre Moulin, and Jie Zhou. Multi-manifold deep metric learning for image set classification. In *CVPR*, pages 1137–1145, 2015.

[66] Dónall A Mac Dónaill. A parity code interpretation of nucleotide alphabet composition. *Chemical Communications*, (18):2062–2063, 2002.

[67] Guillaume Marçais and Carl Kingsford. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics*, 27(6):764–770, 2011.

[68] Darragh G McArt et al. Systematic random sampling of the comet assay. *Mutagenesis*, 24(4):373–8, July 2009.

[69] V. J. McKelvey-Martin, M. H. Green, P. Schmezer, B. L. Pool-Zobel, M. P. De Meo, and A. Collins. The single cell gel electrophoresis assay (comet assay): a european review. *Mutat Res*, 288(1):47–63, 1993.

[70] Marvin Minsky and Seymour Papert. Perceptrons. 1969.

[71] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[72] Radford M Neal. Annealed importance sampling. *Statistics and Computing*, 11(2):125–139, 2001.

[73] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*. Granada, Spain, 2011.

[74] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436, 2015.

[75] Timothy W. Nilsen and Brenton R. Graveley. Expansion of the eukaryotic proteome by alternative splicing. *Nature*, 463(7280):457–463, 2010.

[76] Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *CVPR*, pages 30–38, 2016.

[77] Michiel O Noordewier, Geoffrey G Towell, and Jude W Shavlik. Training knowledge-based neural networks to recognize genes in DNA sequences. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 530–536, 1990.

[78] Peggy L. Olive, Judit P. Banáth, and Ralph E. Durand. Heterogeneity in radiation-induced DNA damage and repair in tumor and normal cells measured using the comet assay. *Radiation Research*, 122(1):86–94, 1990.

[79] O Ostling and KJ Johanson. Microelectrophoretic study of radiation-induced dna damages in individual mammalian cells. *Biochemical and biophysical research communications*, 123(1):291–298, 1984.

[80] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[81] Mihaela Pertea, Xiaoying Lin, and Steven L Salzberg. GeneSplicer: a new computational method for splice site prediction. *Nucleic Acids Research*, 29(5):1185–1190, 2001.

[82] Scott Reed, Kihyuk Sohn, Yuting Zhang, and Honglak Lee. Learning to disentangle factors of variation with manifold interaction. In *ICML*, 2014.

[83] Martin G Reese, Frank H Eeckman, David Kulp, and David Haussler. Improved splice site detection in Genie. *Journal of Computational Biology*, 4(3):311–323, 1997.

[84] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[85] Andreas Rothfuss, Petra Schütz, Sylvia Bochum, Tanja Volm, Elke Eberhardt, Rolf Kreienberg, Walther Vogel, and Günter Speit. Induced micronucleus frequencies in peripheral lymphocytes as a screening test for carriers of a brca1 mutation in breast cancer families. *Cancer Research*, 60(2):390–394, 2000.

[86] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[87] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–6, 2000.

[88] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(9):533–536, 1986.

[89] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[90] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted Boltzmann machines for collaborative filtering. In *ICML*, pages 791–798, 2007.

[91] Matthew B. Schabath, Margaret R. Spitz, H. Barton Grossman, Kerang Zhang, Colin P. Dinney, Ping-Ju Zheng, and Xifeng Wu. Genetic in-

stability in bladder cancer assessed by the comet assay. *Journal of the National Cancer Institute*, 95(7):540–547, 2003.

[92] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[93] Narendra P Singh, Michael T McCoy, Raymond R Tice, and Edward L Schneider. A simple technique for quantitation of low levels of dna damage in individual cells. *Experimental cell research*, 175(1):184–191, 1988.

[94] Sharad Singhal and Lance Wu. Training multilayer perceptrons with the extended Kalman algorithm. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 133–140, 1988.

[95] Nikolai P Sirota, Aliy K Zhanataev, Elena A Kuznetsova, Eugenii P Khizhnyak, Elena A Anisina, and Andrei D Durnev. Some causes of inter-laboratory variation in the results of comet assay. *Mutation Research/Genetic Toxicology and Environmental Mutagenesis*, 770:16–22, 2014.

[96] Soren Sonnenburg, Gabriele Schweikert, Petra Philips, Jonas Behr, and Gunnar Ratsch. Accurate splice site prediction using support vector machines. *BMC Bioinformatics*, 8(Suppl 10):S7, 2007.

[97] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[98] Gary D Stormo, Thomas D Schneider, Larry Gold, and Andrzej Ehrenfeucht. Use of the 'Perceptron' algorithm to distinguish translational initiation sites in E. coli. *Nucleic Acids Research*, 10(9):2997–3011, 1982.

[99] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[100] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[101] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[102] Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *ICML*, pages 1064–1071, 2008.

[103] Vikrant Singh Tomar and Richard C Rose. Manifold regularized deep neural networks. In *Proceedings of InterSpeech*, pages 348–352, 2014.

[104] Cole Trapnell, Lior Pachter, and Steven L. Salzberg. TopHat: discovering splice junctions with RNA-Seq. *Bioinformatics*, 25(9):1105–1011, 2009.

[105] Cole Trapnell, Brian A. Williams, Geo Pertea, Ali Mortazavi, Gordon Kwan, Marijke J. van Baren, Steven L. Salzberg, Barbara J. Wold, and Lior Pachter. Transcript assembly and quantification by RNA-Seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature Biotechnology*, 28(5):511–515, 2010.

[106] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(11):2579–2605, 2008.

[107] Sreenath Rao Vantaram and Eli Saber. Survey of contemporary trends in color image segmentation. *Journal of Electronic Imaging*, 21(4):040901–1–040901–28, 2012.

[108] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *ICML*, pages 1096–1103, 2008.

[109] Li Wan, Matthew D Zeiler, Sixin Zhang, Yann LeCun, and Robert Fergus. Regularization of neural networks using dropconnect. In *ICML*, 2013.

[110] Kai Wang, Darshan Singh, Zhen Zeng, Stephen J. Coleman, Yan Huang, Gleb L. Savich, Xiaping He, Piotr Mieczkowski, Sara A. Grimm, Charles M. Perou, James N. MacLeod, Derek Y. Chiang, Jan F. Prins, and Jinze Liu. MapSplice: accurate mapping of RNA-seq reads for splice junction discovery. *Nucleic Acids Research*, 38(18):e178, 2010.

[111] Yuhao Wang and Jianyang Zeng. Predicting drug-target interactions using restricted Boltzmann machines. *Bioinformatics*, 29(13):126–134, 2013.

[112] Dan Wei, Weiwei Zhuang, Qingshan Jiang, and Yanjie Wei. A new classification method for human gene splice site prediction. In *Health Information Science*, pages 121–130. Springer, 2012.

[113] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory networks. In *ICLR*, 2015.

[114] David K Wood, David M Weingeist, Sangeeta N Bhatia, and Bevin P Engelward. Single cell trapping and DNA damage analysis using microwell arrays. *Proceedings of the National Academy of Sciences of the United States of America*, 107(22):10008–13, June 2010.

[115] Y. Yuan, L. Mou, and X. Lu. Scene recognition by manifold regularized deep learning architecture. *IEEE TNNLS*, 26(10):2222–2233, 2015.

[116] Charles T Zahn and Ralph Z Roskies. Fourier descriptors for plane closed curves. *Computers, IEEE Transactions on*, 100(3):269–281, 1972.

[117] Murizal Zainol et al. Introducing a true internal standard for the Comet assay to minimize intra- and inter-experiment variability in measures of DNA damage and repair. *Nucleic Acids Research*, 37(22):e150, December 2009.

[118] Matthew Zeiler and Robert Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *ICLR*, 2013.

[119] Ya Zhang, Chao-Hsien Chu, Yixin Chen, Hongyuan Zha, and Xiang Ji. Splice site prediction using support vector machines with a Bayes kernel. *Expert Systems with Applications*, 30(1):73–81, 2006.